CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Digital library of Brno University of Technology

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

NÁVRH FIREMNÍ DATABÁZE EVIDENCE SMLUV

DESIGN OF COMPANY CONTRACTS DATABASE

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL KUDA

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ KŘÍŽ, Ph.D.

SUPERVISOR

BRNO 2015

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2014/2015
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Kuda Michal

Manažerská informatika (6209R021)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává bakalářskou práci s názvem:

Návrh firemní databáze evidence smluv

v anglickém jazyce:

Design of Company Contracts Database

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Podle § 60 zákona č. 121/2000 Sb. (autorský zákon) v platném znění, je tato práce "Školním dílem". Využití této práce se řídí právním režimem autorského zákona. Citace povoluje Fakulta podnikatelská Vysokého učení technického v Brně.

Seznam odborné literatury:

BORONCZYK, Tim. PHP 6, MySQL, Apache: vytváříme webové aplikace. Vyd. 1. Brno: Computer Press, 2009, 816 s. ISBN 978-80-251-2767-4.
KOCH, Miloš. Datové a funkční modelování: vytváříme webové aplikace. 4. rozšířené vyd. Brno: Akademické nakladatelství CERM, 2010, 142 s. Učební texty vysokých škol. ISBN 978-80-214-4125-5.
LACKO, Ľuboslav. Mistrovství v SQL Server 2012. 1. vyd. Brno: Computer Press, 2013, 640 s. ISBN 978-80-251-3773-4.
OPPEL, Andrew J. SQL bez předchozích znalostí: [průvodce pro samouky]. Vyd. 1. Brno: Computer Press, 2008, 240 s. Učební texty vysokých škol. ISBN 978-80-251-1707-1.
PROCHÁZKA, David. PHP 6: začínáme programovat. 1. vyd. Praha: Grada, 2012, 183 s. Průvodce (Grada). ISBN 978-80-247-3899-4.

Vedoucí bakalářské práce: Ing. Jiří Kříž, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2014/2015.

L.S.

doc. RNDr. Bedřich Půža, CSc. Ředitel ústavu	doc. Ing. et Ing. Stanislav Škapa, Ph.D. Děkan fakulty

V Brně, dne 28.2.2015

Abstrakt

Práce pojednává o problematice výběru, návrhu a implementace vhodného databázového systému. Zabývá se historií jejich vývoje a vybrané systémy také porovnává. V mojí práci je popsán návrh datového modelu pro firemní databázi smluv a nastíněno řešení jejího aplikačního rozhraní.

Abstract

This bachelor's thesis deals with the issue of suitable database system selection, design and implementation. It contents the historical development overview and it's comparing. My publication describes design of company contracts database and outlines the application programming interface.

Klíčová slova

Databáze, návrh databáze, tvorba databáze, databázový model, relace, integrita, normalizace, ER digram, SQL, MSSQL server, Oracle, PHP, Apache, aplikace

Key words:

Database, database design, database creation, database model, relation, integrity, normalization, ER diagram, SQL, MSSQL server, Oracle, Apache, application

Bibliografická citace mé práce:

KUDA, M. *Návrh firemní databáze evidence smluv*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2015. 69 s. Vedoucí bakalářské práce Ing. Jiří Kříž, Ph.D..

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil autorská práva (ve smyslu zákona č. 121/2000 Sb. o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 5. 5. 2012

Michal Kuda

Tímto bych chtěl poděkovat panu Ing. Jiřímu Křížovi, Ph.D. za odborné vedení bakalářské práce a čas, mně věnoval. Dále také vedení a zaměstnancům společnosti Ready Control s.r.o. za to, za poskytnuté podklady a informace.

Obsah

Úvod

V dnešní době nabývají informační technologie na významu a jejich prorůstání do administrativy a řízení firem, společností, státních institucí a mnohých dalších subjektů je stále masivnější. Především jde o zpracovávání masivních objemů dat a s tím souvisí rozvoj podnikových informačních systémů a databázových systémů, které stávají pomocníky nejen velkých firem, ale i drobným podnikatelům, jimž slouží k uchovávání informací v podobě dat, a jejich následnému získávaní poznatků z těchto dat prostřednictvím filtrů, dotazů nebo různých datových analýz. V současnosti je uchovávání dat v podobě papírových kartoték, byť se s nimi stále můžeme setkávat, přežitkem, a proto se databázové systémy a jejich programovací jazyky dostávají stále více do povědomí i širší veřejnosti. Hlavním smyslem a cílem databázových systémů je schopnost data neustále aktualizovat, analyzovat a získávat z nich konkrétní poznatky, které jsou používány v rozhodovacích procesech managementu firem.

S vývojem databázových systémů se rozšiřuje množství platforem, na kterých běží, a současně se výrazně snižuje jejich pořizovací cena. To umožňuje i malým podnikatelským subjektům používat tyto databázové systémy v širším měřítku. Rovněž si mohou vybrat ze nemalé škály databází, jako jsou například transakční databáze, kdy podnikatel neustále potřebuje aktualizovat informace o různých výrobcích a jejich stavu, pohybu smluv atd.. Nebo datové sklady sloužící pro rozsáhlé a dlouhodobé uchovávání dat pro provádění rozsáhlých složitých analýz, které jsou východiskem pro mnoho zkušených manažerů řídících obrovské podniky.

Některé starší databáze již nemusí splňovat požadavky kladené uživateli a současnými uživatelskými procesy. Proto je v důsledku modernizace nutné neustále databázové systémy vyvíjet a přizpůsobovat je uživatelským požadavkům. Ovšem množství platforem a systémů může být pro nezkušeného uživatele nepřehledné a složité. Proto potřebují poradit nebo pomoci s výběrem a implementací databázového systému do firmy. Tímto problémem se budu zabývat ve své bakalářské práci, která bude pojednávat o výběru databázového systému a následnému vytvoření databáze smluv, které zjednoduší administrativu a zprůhlední vznik a zánik smluv a následnému spravedlivému hodnocení a vyplácení odměn za existující smlouvy. V současné době firma za tímto účelem žádný databázový systém nevyužívá.

Vymezení problému

Problém, před který jsem byl postaven, mě přivedl na teorii relačních databází, které jsou v dnešní době velmi populární a velmi rozšířené. Jde především o rozpolcenost firmy při využívání informačních technologií. V některých segmentech dokonce nejsou informační technologie využívány vůbec nebo jen velmi sporadicky. Situace ve firmě se stala neudržitelnou a pomalu začala jednotlivé agendy převádět do datové informační podoby. Ať už účetnictví, evidence firemního majetku, zaměstnanců a podobně. Já ve firmě působím externě v rámci studia a byl jsem manažerem firmy požádán o vytvoření databáze smluv, která v tom okamžiku existovala pouze v excelovské podobě. Zmiňované zpracování agendy smluv je krajně nepraktické až téměř nepoužitelné, protože globální aktualizace byla velmi komplikovaná, distribuce databáze rovněž tak. Duplicitní údaje byly snadno přehlédnutelné, čehož následkem byly výdaje navíc, ovšem neoprávněné. Navíc tento systém zpracování dat znemožňoval efektivní kontrolu, celkové souhrny a součty jednotlivých smluv a jejich pořizovatelů a v neposlední řadě i zpracování měsíčních a ročních statistik a výsledků hospodaření podniku. Jinými slovy museli pracovníci back office skládat údaje dohromady z papírových materiálů, excelovských tabulek a výkazů některých softwarových programů.

Cíl práce

Cílem mé bakalářské práce je výběr databázového systému a vytvoření návrhu databáze, která bude řešit nedostatky v evidenci smluv. V mém případě půjde o návrh transakční databáze na platformě Microsoft SQL server 2012. V první části práce se zaměřím na teoretickou problematiku databází obecně, jejich vznik a vývoj do dnešní doby a k čemu jsou dobré. Následně se budu věnovat relačním databázím. Následně rozeberu podrobněji současnou situaci a problematiku firmy. A nakonec bude následovat praktické řešení dané situace. Tedy vlastní návrh a tvorba relační databáze, které budou podrobně popsány.

1. Teoretická východiska

1.1. Historie databázových systémů

Již v roce 1890 Herman Hollerith první automat na bázi děrných štítků a v roce 1911 došlo ke spojení jeho firmy s další a vzniku International Business Machines, tedy IBM. V roce 1935 byla ve spojených státech uzákoněna nutnost vedení informací o cca 26 miliónech zaměstnanců, a proto IBM nové zařízení. Byl to první digitální počítač pro komerční využití *UNIVAC I*. V roce 1959 bylo pro Pentagon vyrobeno již 200 těchto počítačů, jejichž mozkem byl děrný štítek. V roce 1960 vzniklo seskupení Data Systems Languages (Codasyl), ustavené ministerstvem obrany USA pro standardizaci softwarových aplikací a výsledkem byl common business – oriented language neboli *COBOL*. Neméně důležitým pokrokem byl přechod od magnetických pásek, které umožňovali jen sériový přístup k datům (což značně stěžovalo vyhledávání dat a možnost definice nějakého efektivnějšího databázového modelu), k magnetickým diskům. V roce 1961 Charles Bachman z General Elelctric představil první integrovaný datový sklad s prvním náznakem databázového managementu a jinými vlastnostmi. Později v šedesátých letech pak Bachman a další výzkumníci založili v rámci uskupení Codasyl samostatnou *Database Task Group*. Skupina publikovala základní specifikaci pro programovací jazyky, zejména *COBOL*, určené pro práci s databázemi. Codasyl produkt měla i IBM, uvedla ho v roce 1968 pod názvem IMS. Většina Codasyl kompatibilních databází používala síťový model dat, zatímco IBM u své implementace použila hierarchický model (o databázových modelech si něco povíme v následujících kapitolách). V roce 1970 publikoval absolvent Oxfordu *Edgar Frank Codd*, který vstoupil do IBM v roce 1949, článek „A Relational Model of Data for Large Shared Data Banks“, což byl návrh na implementaci nového datového modelu, a to modelu *„relačního“*.

Codd načrtl možnost, jak použít relační kalkul a algebru i pro netechnické uživatele při ukládání a manipulaci s daty. Dle jeho pojetí k tomu mělo být používáno srozumitelných příkazů vycházejících z běžné angličtiny. Už tato původní koncepce předpokládala ukládání dat do tabulek. Tento návrh založen na nezávislosti dat na použitém hardwaru, na způsobu jejich fyzického uložení, dále pak na přístupu k datům pomocí neprocedurálního jazyka. Uživatel měl mít možnost specifikovat operaci nad

jím vybranou množinou dat namísto pouhé manipulace s jedním záznamem. Zpočátku byla IBM k tomuto návrhu skeptická a nechtěla se vrhnout do něčeho, co by ve své podstatě zavrhlo její produkt IMS. Nakonec po různých debatách vznikly dva projekty relačních databází, a to *System-R* v rámci IBM a od roku 1972 projekt *Ingres* na University of California at Berkeley. Původní jazyk System-R byl *SEQUEL* (Structured English Query Language) a v listopadu roku 1976 byla v IBM Journal of R&D popsána verze *SEQUEL2*, které později přejmenovali na *SQL*. Ingres a System-R byly oba vyvíjeny pod různými operačními systémy a Ingres používal místo jazyka *SQL* vlastní nicméně velmi podobný jazyk *QUEL*. Postupně byl Ingres uvolněn přibližně v tisícovce kopií a profesor Michael Stonebraker založil Ingres Corporation za účelem kód Ingresu komercionalizovat. Můžeme říci, že kód z Berkeley se stal inspirací nebo přímo zdrojem k založení téměř všech dnes známých SQL databází.

V roce 1980 na trhu vyšla prní SQL databáze pro počítače VAX a byla to databáze *ORACLE*. Po tomto počinu firmy Oracle přišla na trh se svým řešením i IBM, jejich produkt byl určen pro počítače MVS a jmenoval se *DB2*. Osmdesátá léta můžeme považovat za počátek zlatého věku relačních databází. Projekt Ingres byl v roce 1985 transformován do projektu *POSTGRES*, jehož ambicí bylo vytvořit relačně – objektovou databázi. Objektově orientovaná databáze umožňuje definovat uživatelům vlastní metody, pomocí kterých se přistupuje a manipuluje s daty. Postupně Postgres přechází do open source podoby a v roce 1996 je přejmenován na *PostgreSQL* a je vyvíjen dodnes. SQL databáze nejsou odvětvím, kde by byl vývoj takový, že každý den vyjde nová verze databáze a její nové vlastnosti. Jak můžeme vidět z historie, vývoj podobných projektů trvá 10 – 15 let. Samotné dnes existující standardy představují dostatečnou inspiraci v tom, co dál, protože ne vše z těchto standardů je již podporováno a i mezi jednotlivými producenty databází existují rozdíly v úrovni podpory těchto standardů.

1.2. Přehled vybraných databázových systémů současnosti

Po krátkém pohledu do historie databázových systémů si teď některé z nich stručně představíme. K dispozici jsou jak komerční produkty, tak i produkty distribuované zdarma jako open source. Každý program má svá specifika, a proto jsou produkty pro

různé typy úloh různě vhodné. Respektive jsou určitý typ úloh vhodné, srovnatelné, jako některé další systémy, a nebo jsou nevhodné. Všechny databázové systémy, které zde zmíním, jsou potencionálně vhodné pro dosažení cíle této práce. Nakonec z nich vyberu jeden, ve kterém vytvořím návrh databáze.

1.2.1. Ingres II

Ingres II byl vytvořen na Univesity of California at Berkeley, z čehož je zřejmé, že je příbuzným systémem s PostgreSQL. Dříve byl distribuván jako komerční produkt, ale nyní je dostupný jako open source.

1.2.2. mSQL

Vznikl v roce 1994. Do této doby nebyl na trhu produkt, který by zaplnil místo mezi databázemi typu Microsoft Access a databázovými systémy typu Oracle nebo DB2. Systém byl volbou pro programátory, kteří nechtěli investovat nemalé finanční prostředky do komerčních produktů. Přestože nebyly uvolněny zdrojové kódy, bylo možné databázový systém využívat bez omezení a libovolně. Postupně jej ale začal vytlačovat jiný systém, a to MySQL, který vycházel ze stejné architektury.

1.2.3. Berkeley

Tento systém, tedy Oracle Berkeley DB, vlastní firma Oracle Corporation, ale byl vyvinut na University of California at Berkeley a patří do skupiny open source databází. Systém Berkeley umožňuje vývojářům implementovat do jejich aplikací škálovatelný a rychlý transakční databázový engine. Výsledkem toho je, že se k zákazníkům a koncovým uživatelům dostává aplikace, pracuje velmi jednoduše, spolehlivě řídí data, zvládá i nadměrná zatížení, a také nevyžaduje žádnou pokročilou a komplikovanou správu. Data jsou ukládána do jednotlivých souborů, kde každé jednotlivé databázi odpovídá jeden soubor.

1.2.4. MSSQL

Microsoft SQL server je relační systém řízení báze dat a rovněž jako firma Oracle je i Microsoft klíčovým hráčem na trhu. Postupně se jeho kvalita blíží top produktům na trhu, nicméně jeho pořizovací cena a kompatibilita pouze s operačním systémem Microsoft Windows, což ještě navyšuje pořizovací náklady, ho poněkud handicapují. MS SQL splňuje požadavky normy ANSI SQL – 92 a i tento systém ji rozšiřuje. Ne však do té míry jako některé jiné systémy.

1.2.5. Firebird

„*InterBase je mocný, kompaktní relační databázový systém na bázi klient/server, provozovaný na celé škále operačních systémů včetně Microsoft Windows, Linuxu a řadě dalších UNIXových systémů a nabízející vyšší úroveň podpory standardů jazyka SQL než většina dostupných databázových systémů*“ píše odborník na Firebird Pavel Císař v úvodu své knihy. V létě roku 2000 uvolnila společnost Borland zdrojové texty systému InterBase, na kterém je postaven právě relační databázový systém Firebird. Firebird byl vyvíjen v rámci stejnojmenného open source projektu, za jehož financováním stála nezisková organizace FirebirdSQL Foundation a sdružení dalších zainteresovaných firem. Tento produkt se stal velmi populární zejména v Rusku. Jedná se již o velmi komplexní a plnohodnotný databázový systém.

1.2.6. Oracle

Oracle je jeden z nejlepších databázových systémů na světě vůbec. Byl vyvíjen již od roku 1978 ve firmě Oracle Corporation, která byla založena přeběhlíky z univerzitního výzkumu v americkém Berkeley. Systém Oracle bez problémů běží jak na obyčejných PC s operačním systémem Microsoft Windows, tak i na řadě distribucí operačních systému z rodiny UNIX a samozřejmě i na multiprocesorových počítačích. Pokud běží aplikace na odpovídajícím hardwaru, je schopna s přehledem zpracovávat až miliardy řádků dat, což ho předurčuje pro ty nejrozsáhlejší projekty v nejnáročnějších oblastech, například ve státní správě a podobně. Oracle Database je kompletní platforma pro ukládání, správu a analýzu dat. Jedinečná podpora produktů, unikátním způsobem

řešená bezpečnost, výkonnost, ukládání dat v kombinaci s dalšími vlastnostmi z Oracle dělají jednu z nejspolehlivějších databázových systémů. Na druhou stranu má ale Oracle velmi vysokou pořizovací cenu (můžeme si samozřejmě vybrat z více variant), náročnou instalaci a správu. Proto je tento systém implementován jen u opravdu velkých projektů.

1.2.7. MySQL

MySQL je produktem švédské firmy MySQL AB. Jedná se o multiplatformní atabázový systém a jeho první verze byla vydána v roce 1995. Existuje sice komerční verze distribuce, ale jde především o volně šiřitelný software. Nejen díky tomu je ale My SQL zřejmě nejrozšířenějším databázovým systémem, ale i díky velmi vysokému výkonu, a také díky spojení LAMP, tedy spojení databáze s webovým serverem Apache, komunikace se skripty programovacího jazyka PHP, přičemž vše běží na operačním systému LINUX, používané jako platforma pro implementaci dynamických webových stránek.

Mimo jiné vlastnosti se MySQL vyznačuje:

- Podporou multiprocesorových počítačů
- Až 32 indexů na jednu tabulku
- Velmi dobře propracovaná dokumentace a velká podpora veřejností
- Snadné získání informací o stavu databáze
- Propracované funkce pro matematické kalkulace a třídění dat
- Datové typy jsou schopny pokrýt prakticky všechny druhy dat

1.2.8. SQLite

Tato platforma je vhodná opravdu jen na ty nejmenší projekty. Nejedná se o kompletní databázový systém jako jsou Oracle, MySQL, MSSQL či PostgreSQL, ale je to pouze malá knihovna napsaná v jazyce C v podstatě pouze implementuje jazyk SQL nad souborem dbm. Celá databáze je umístěna v jediném souboru na disku, nicméně může

být sdílen mezi různými platformami i počítači. Pokud se s databází nepracuje, neběží v systému žádný proces, SQLite se tedy nechová jako server. Její síla je v jednoduchosti, velmi slušném výkonu, nízké náročnosti na systémové zdroje a poměrně kompletní implementaci jazyka SQL. Další vlastností SQLite, která stojí za zmínku, je její velikost, která činí přibližně 350kB. Ze standardu SQL92 toho umí poměrně mnoho, i když to velmi často bývá nějakým způsobem omezeno. Například nezná datové typy a velikost jednoho záznamu nesmí přesáhnout 1MB.

1.2.9. PostgreSQL

Multiplatformní databázový systém PostgreSQL je pokračovatelem systému POSTGRES, který, jak jsme zmiňovali v kapitole o historii DB systémů, byl vytvořen v letech 1977-1985 týmem profesora Stonebrakera na UC v Berkeley. Po ukončení projektu ho opět vzkřísili dva postgraduální studenti, za což jim patří velký dík. Systém PostgreSQL se stal velmi populárním a vychvalovaným tisíci spokojených uživatelů. Především pro excelentně zpracovanou dokumentaci, vysokou spolehlivost a bezpečnost a další užitečné vlastnosti.

Kromě všech běžných funkcí nabízí PostgreSQL i nadstandardní funkce jako:

- Informační schémata
- Podporu více procesů
- Dvoufázové potvrzování dotazů
- Částečné a funkcionální indexy

1.3. Databázové modely

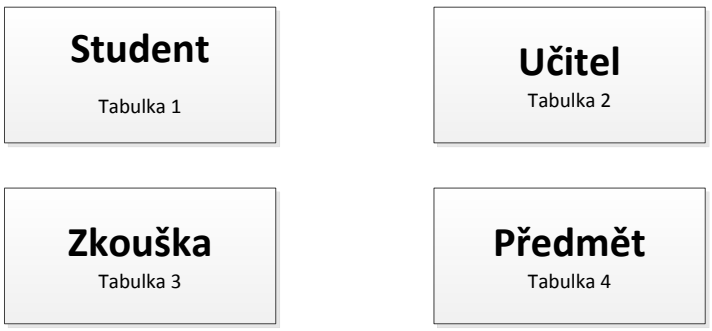
Pokud vytváříme informační systém, potom zpravidla nevystačíme s jedinou strukturou věty (věta je jeden záznam v tabulce, jinými slovy celý jeden řádek v dané tabulce). Pro jednotlivé typy datového objektu potřebujeme vytvořit návrh samostatné datové struktury věty. Pokud budeme například budovat jednoduchý informační systém posluchačů vysoké školy, budeme potřebovat navrhnout datové struktury přinejmenším pro objekty student, zkouška, předmět a učitel. [6]

V rámci informačního modelu se snažíme vytvořit odpovídající obraz reality. A to tak, aby data vložená do systému této realitě plně odpovídala. Například, že konkrétního studenta zkoušel z konkrétního předmětu konkrétní učitel s konkrétním hodnocením. [6]

V současné době může projektant informačních systémů vybírat z několika modelů:

- lineární
- hierarchický
- síťový
- relační
- objektový

1.3.1. Lineární model

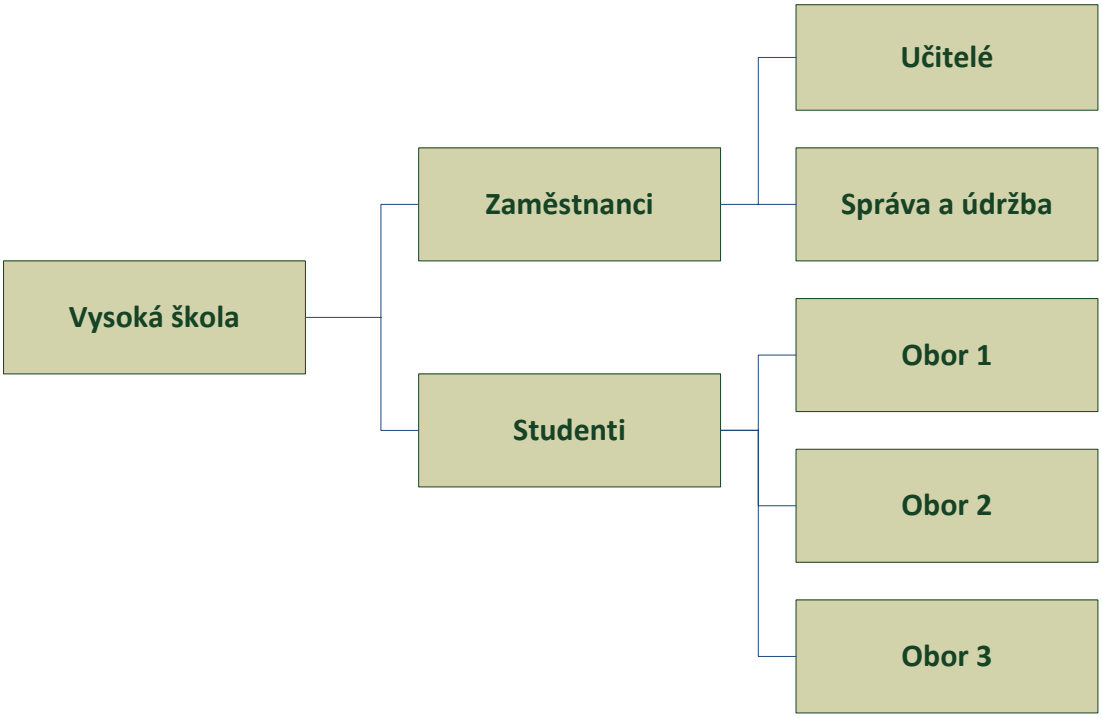


Obrázek 1: *Lineární datový model*
Zdroj [6]

Lineární datový model je zobrazen na obrázku výše, kde každý obdélník představuje jeden soubor s větami v příslušné struktuře (s příslušnými atributy). Pokud použijeme na realizaci tohoto modelu databázový systém, každý obdélník bude tvořit jednu tabulku databáze. Mezi jednotlivými skupinami objektů – tabulkami v lineárních databázových modelech není žádná vazba. Je to jediný datový model, který můžeme implementovat na libovolném médiu. [6]

1.3.2. Hierarchický model

V hierarchickém modelu jsou data organizována do stromové struktury. Každý záznam představuje uzel ve stromové struktuře a vzájemný vztah mezi záznamy je typu rodič/potomek. Proto nalezení dat v hierarchické databázi vyžaduje navigaci přes záznamy směrem na potomka, zpět na rodiče nebo do strany na dalšího potomka. Zásadními nevýhodami hierarchického uspořádání je složitá operace vkládání a rušení záznamů, a v některých případech i nepřirozená organizace dat.



Obrázek 2: *Hierarchický datový model*
zdroj: Vlastní

1.3.3. Síťový model

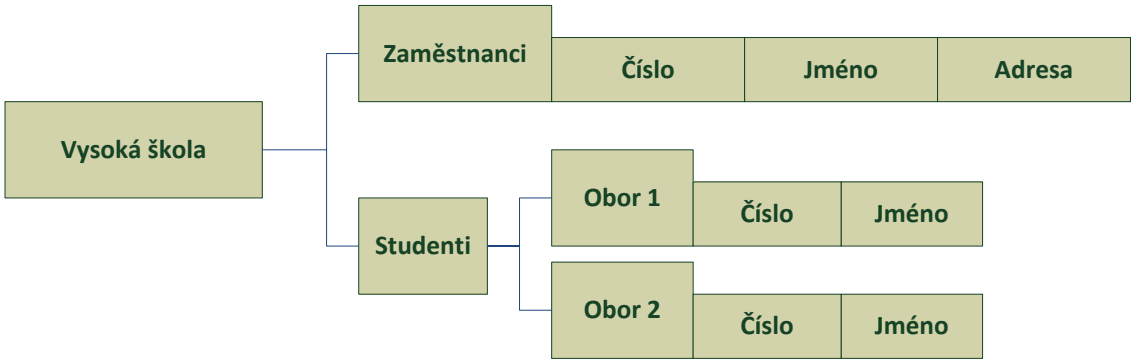
Síťový model dat je v podstatě zobecněním hierarchického modelu, který doplňuje o mnohonásobné vztahy. Tyto vztahy, nebo-li sety propojují záznamy různého či stejného typu, přičemž spojení může být realizováno na jeden nebo více záznamů. Přístup k propojeným záznamům je přímý bez dalšího vyhledávání.

K dispozici jsou operace:

- nalezení záznamu podle klíče

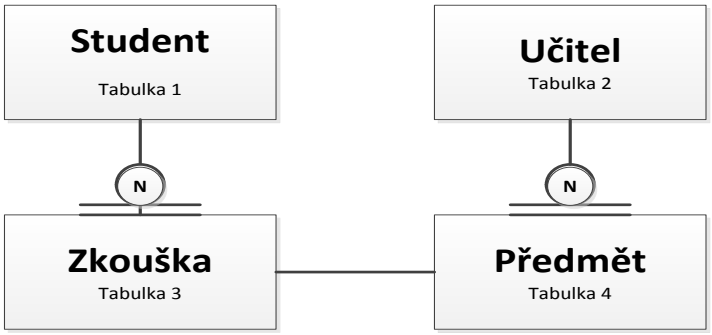
- posun na prvního potomka v dílčím setu
- posun stranou na dalšího potomka v setu
- posun nahoru z potomka na jeho rodiče v jiném setu

Nevýhodou síťové databáze je zejména nepružnost a obtížná změna její struktury.



Obrázek 3: *Síťový datový mode*
Zdroj: Vlastní

1.3.4. Relační model



Obrázek 4: *Relační datový model*
Zdroj: [6]

V současnosti je nejpoužívanějším datovým modelem relační datový model, resp. objektově – relační model. Vzniká propojením více lineárních modelů spojených dohromady prostřednictvím položek, kterým říkáme *relační klíče*. Toto spojení vzniká v okamžiku, kdy potřebujeme mít společně k dispozici data ze všech spojených tabulek, a zaniká, v okamžiku, kdy práci s modelem ukončíme. Jednotlivé lineární modely lze pochopitelně využívat i samostatně. [6]

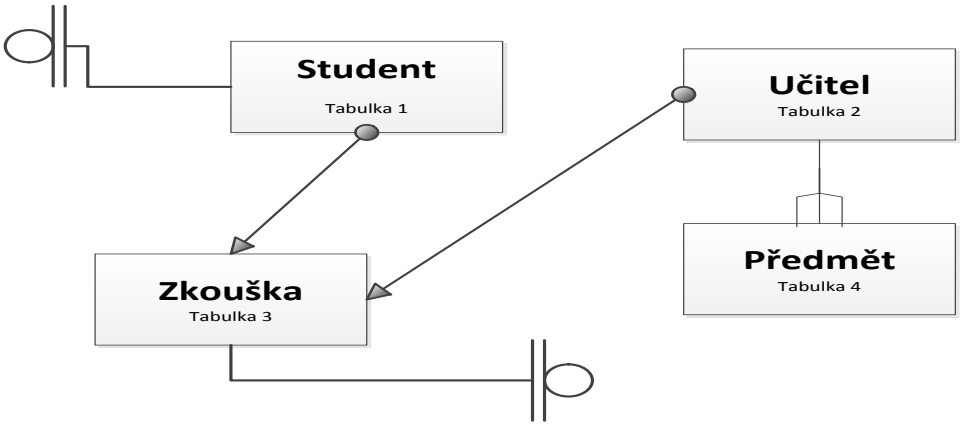
Dr. Codd definoval 12 základních pravidel pro relační systém řízení báze dat:

- *Informační pravidlo* – všechny informace v relační databázi jsou vyjádřeny explicitně na logické úrovni jediným způsobem – hodnotami v tabulkách
- *Pravidlo jistoty* – všechna data v relační databázi jsou zaručeně přístupná kombinací jména tabulky s hodnotami primárního klíče a jménem sloupce
- *Systematické zpracování nulových hodnot* – nulové hodnoty jsou plně podporovány relačním SŘBD pro reprezentaci informace, která není definována a to nezávisle na datovém typu
- *Dynamický on-line založený na relačním modelu* – popis databáze je vyjádřen na logické úrovni stejným způsobem jako zákaznická data, takže autorizovaný uživatel může aplikovat stejný relační jazyk ke svému dotazu jako uživatel při práci s daty
- *Obsáhlý datový podjazyk* – relační systém může podporovat několik jazyků a různých módů použitých při provozu terminálu. Nicméně musí být nejméně jeden příkazový jazyk s dobře definovanou syntaxí, který obsáhle podporuje definici dat, definici pohledů, manipulaci s daty jak interaktivně, tak programem, integritní omezení, autorizovaný přístup k databázi, transakční příkazy apod.
- *Pravidlo vytvoření pohledů* – všechny pohledy, které jsou teoreticky možné, jsou také systémem vytvořitelné
- *Schopnost vkládání, vytvoření a mazání* – schopnost zachování relačních pravidel u základních i odvozených relací je zachována nejen při pohledu na data, ale i při operacích průniku, přidání a mazání dat
- *Fyzická datová nezávislost* – aplikační programy jsou nezávislé na fyzické datové struktuře
- *Logická datová nezávislost* – aplikační programy jsou nezávislé na změnách v logické struktuře databázového souboru
- *Integritní nezávislost* – integritní omezení se musí dát definovat prostředky relační databáze nebo jejím jazykem a musí být schopna uložení v katalogu a nikoliv v aplikačním programu
- *Nezávislost distribuce* – relační SŘBD musí být schopny implementace na jiných počítačových architekturách

- *Pravidlo přístupu do databáze* – jestliže má relační systém jazyk nízké úrovně, pak tato úroveň nemůže být použita k vytváření integritních omezení a je nutno vyjádřit se v relačním jazyce vyšší úrovně

1.3.5. Objektový model

Nejnovějším datovým modelem je objektový datový model. Objektové datové modely jsou vystavěny na základním prvku – objektu, který odpovídá přibližně pojmu věta, a dále má kromě svých atributů i definované metody, které determinují chování objektu.



Obrázek 5: *Objektový datový model*
Zdroj: [6]

Mějme takový objekt „*zkouška*“ studenta a atributy objektu zkouška budou datum, známka, číslo studenta, termín, předmět a zkoušející. Dále můžeme na objektu zkouška nadefinovat metody, jako například „*vytvoř záznam o zkoušce*“, která ověří , zda daný student nemá již tuto zkoušku hotovou, či zda nemá vyčerpané termíny.

Objekty stejného typu tvoří třídu objektů a konkrétní záznam daného objektu nazýváme jeho instancí. Každému objektu v databázi je přidělen jedinečný – unikátní – identifikátor – OID. Jeho prostřednictvím můžeme mezi objekty vést přímé vazby jako

například v síťovém modelu a navíc mohou v objektovém modelu existovat vazby relační.

Základním filozofickým rysem objektových modelů je tzv. „zapouzdření objektu“, což znamená, že jediným způsobem, jak s objektem pracovat je volání některé z metod objektu. A tím docilujeme vysoké datové abstrakce a nezávislosti dat. [6]

1.4. Relační datový model

Relační datové modely jsou založeny na teorii relací. Zjednodušeně můžeme říci, že prostřednictvím relačních datových modelů zachycujeme jak *data* o zkoumaných objektech, tak i *vzájemné vztahy* těchto objektů. To nám umožňuje zachytit svět v co možná nejreálnější podobě. [6]

1.4.1. Relace

Nejdříve si řekneme něco o terminologii, a to z pohledu teorie relací:

- *Schéma relace* – struktura věty (pořadí atributů relace)
- *Atributy relace* – názvy jednotlivých atributů (sloupců) relace
- *N–tice relace* – n–tá věta (řádek) relace
- *Hodnota atributu* – konkrétní hodnota atributu příslušné věty, tedy hodnota konkrétní buňky v tabulce

Pro definování a popis prvků datového modelu se v datovém modelování využívá jako nástroj teorie množin.

Množiny můžeme definovat pomocí domén, neboli prvků množiny:

- výčtem prvků
- logickým vymezením
- sémantickým vymezením

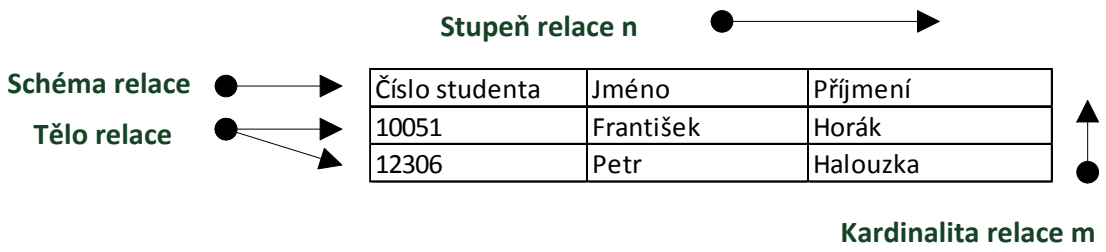
V rámci množin rozlišujeme dva druhy vztahů, a to vztahy mezi prvky množiny a vztahy mezi množinami.

Pro práci s množinami používáme tyto nástroje:

- sjednocení

- průnik
- rovnost
- inkluze
- doplněk

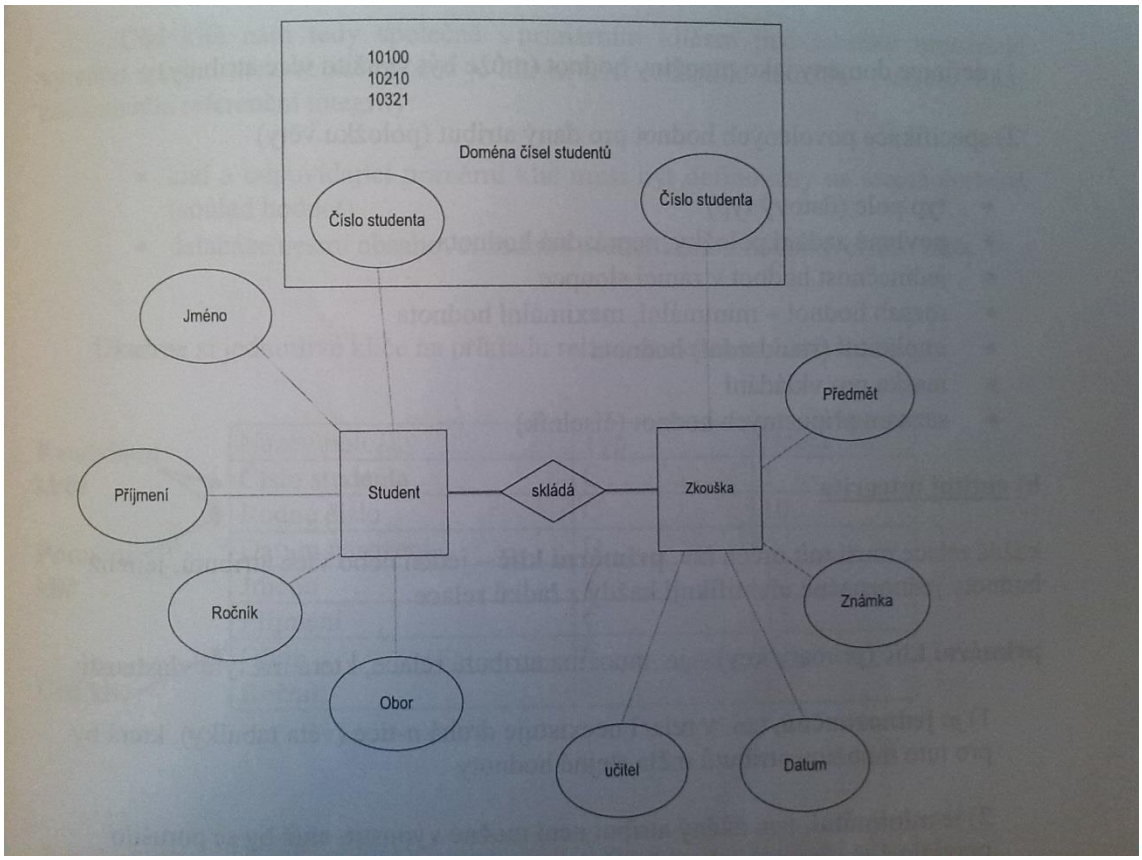
Tělo relace představuje podmnožinu kartézského součinu, například:



Obrázek 6: *Relace*
Zdroj: [6]

Doc. Ing. Miloš Koch a Ing. Bernard Neuwirth uvádějí ve své knize *Datové a funkční modelování* tato pravidla pro tabulkovou prezentaci relace:

- každý řádek odpovídá jedné n-tici relace
- pořadí řádků je nevýznamné
- žádné dva řádky nejsou stejné
- pořadí sloupců je nevýznamné
- význam každého sloupce je určen jménem atributu
- žádné dva názvy sloupců (atributy) nejsou stejné
- hodnoty ve sloupcích jsou atomické [6]



Obrázek 7: *Ukázka schématu relací a jejich vztahu*
Zdroj: [6]

1.4.2. Integrita relačního modelu

Pokud modelujeme data z reálného světa, musíme vzít v úvahu určitá omezení teoretického modelu. Mluvíme o tzv. integritě modelu.

1.4.2.1. Doménová integrita

Doménová integrita neboli integrita hodnot znamená, že každá hodnota každého z atributů relace musí být z množiny přípustných hodnot pro daný atribut:

- datový typ
- povinné zadání položky, neprázdná hodnota
- jedinečnost hodnot v rámci sloupce
- rozsah hodnot – od minimální po maximální hodnotu
- implicitní hodnota
- maska (formát) pro vkládání

- seznam přípustných hodnot – číselník [6]

1.4.2.2. Entitní integrita

Entitní integrita se věnuje problematice tzv. klíčů. Každé relaci přiřazujeme nebo určujeme jeden nebo více atributů, kdy jejich hodnoty jednoznačně identifikují každý z řádků (vět) v tabulce. Jde o tzv. *primární klíč*. Jde o množinu atributů, které mají tyto vlastnosti:

- je *jednoznačná*, tzn. v relaci již neexistuje další taková n-tice, která by pro tuto množinu atributů měla stejné hodnoty
- je *minimální*, tzn. není možné vypustit žádný atribut, aniž by nedošlo k porušení pravidla 1

Takže primární klíč je základním prostředkem adresace n-tic relace a platí pro něj tato pravidla:

- u žádného atributu primárního klíče nesmí chybět hodnota
- a jak plyne z pravidel výše, každá n-tice musí být v každém okamžiku identifikovatelná hodnotou primárního klíče

Výše zmíněné podmínky může splňovat více klíčů (kombinací atributů), které nazýváme *kandidátní klíče*. Z kandidátních klíčů vybereme jeden jako primární a ostatní pak nazýváme *alternativní*. [6]

1.4.2.3. Referenční integrita

Referenční integritu zajišťuje tzv. *cizí klíč*, který musí splňovat tyto nezávislé vlastnosti:

- každá hodnota je buď plně zadaná, nebo plně nezadaná
- existuje jiná relace s takovým primárním klíčem, že každá zadaná hodnota cizího klíče je identická s hodnotou primárního klíče nějaké n-tice této jiné relace

Můžeme tedy pomocí primárního klíče jiné tabulky a cizího klíče vytvářet vztahy mezi relacemi. To je hlavní účel relačního datového modelu a musí pro něj platit pravidla referenční integrity:

- cizí i odpovídající primární klíč musí být definovány na stejné doméně
- databáze nesmí obsahovat žádnou nesouhlasnou hodnotu cizího klíče [6]

1.4.3. Kardinalita vztahů

Kardinalitu vztahů omezuje integritní omezení na 1:1, 1:N, N:1 a M:N. Nejsnáze si jejich kardinalitu přiblížíme na příkladech.

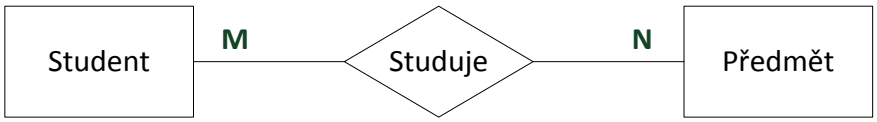
- **1:1 –** říká nám, že vždy jedné větě jedné relace odpovídá jedna nebo žádná věta jiné relace. Mějme například vztah mezi datovými objekty „člověk“ a „řidičský průkaz“. Jeden člověk může vlastnit pouze jeden řidičský průkaz

Obrázek 8: *Vztah 1:1*
Zdroj: [6]

- **1:N**, **N:1 –** říká nám, jedné větě jedné relace odpovídá jedna nebo více vět jiné relace. Vezměme například vztah mezi entitami „student“ a „zkouška“. Jeden student může skládat více zkoušek (což se běžně děje). Vztah N:1 je totéž, jen na něj pohlížíme z opačné strany

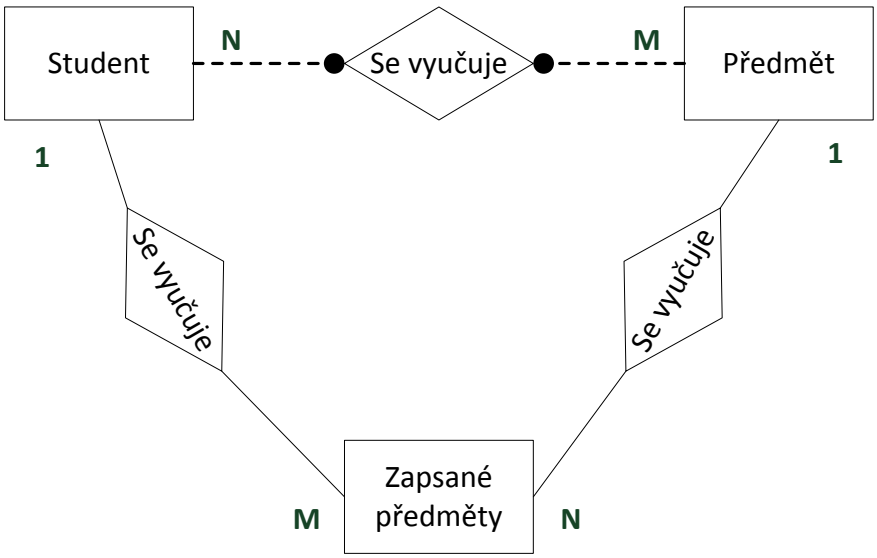
Obrázek 9: *Vztah 1:N*
Zdroj: [6]

- **M:N –** říká nám, že několika větám jedné relace odpovídá jedna nebo více vět jiné relace. Vezměme jako příklad vztah mezi entitami „student“ a „předmět“. Jeden student si zaregistruje více předmětů a naopak jeden předmět je navštěvován více studenty.



Obrázek 10: *Vztah M:N*
Zdroj: [6]

U vztahu M:N můžeme jednoduše stanovit primární klíče v obou tabulkách, ale i přesto nám logika vazby M:N neumožní vést vazbu mezi oběma entitami. Řešení vazby M:N nejdeme ve vytvoření nové entity, v tomto případě „zapsané předměty studenta“ s primárním klíčem složeným z obou primárních klíčů původních entit. Provedli jsme tzv. dekompozici vztahu M:N a novou entitu nazýváme *průniková entita*. [6]



Obrázek 11: *Dekompozice vztahu M:N pomocí průnikové entity*
Zdroj: [6]

1.4.4. Normalizace

Normalizací nazýváme činnost, kterou upravujeme návrhy datových struktur podle určitých normalizačních forem. Tyto formy vychází z požadavku na efektivní ukládání dat a minimalizace redundance při zachování integrity a konzistence dat.

Normalizace je v podstatě postupná dekompozice relací a jejich vztahů do optimálního tvaru. A to tak, aby:

- byla zachována bezztrátovost při zpětném spojení
- byly zachovány závislosti
- bylo odstraněno opakování informací, tzv. redundance dat [6]

1.4.5. 1. normální forma – multizávislost

V podstatě jde o to, že všechny vícehodnotové či složené atributy musíme rozložit v rámci jedné relace na atributy jednoduché nebo, pokud to není možné, dekomponovat na atributy jedné relace na jednoduché atributy ve více relacích. Také jim říkáme *atomické*. [6]

1.4.6. 2. normální forma – funkční závislost

O relaci můžeme říci, že je v druhé normální formě v případě, že je v první normální formě a současně všechny její atributy závisí na celém kandidátním a z něho vybraném primárním klíči. V podstatě se jedná o odstranění všech vazeb typu M:N.

1.4.7. 3. normální forma – tranzitivní závislost

Ve třetí normální formě musí být relace ve druhé normální formě a následně odstraníme všechny *tranzitivní závislosti*. Tzn., že každý neklíčový atribut musí být funkčně závislý na klíči. Pokud na něm závisí přes jiný neklíčový atribut, potom říkáme, že tento atribut je na klíči tranzitivně závislý. [6]

1.4.8. Další normalizační formy

Existují ještě další formy normalizace, které ještě dále zlepšují optimalizaci dat v datovém modelu. Nicméně nejdůležitější jsou právě první tři normální formy. Dalšími jsou například *Boyce – Coddova normální forma, 4. normální forma a 5. normální forma*.

1.5. Návrh databáze

Procesem návrhu databáze je strukturovaný přístup s využitím technik, různých nástrojů, a také dokumentace. Metodologie návrhu databáze má tyto tři fáze:

- konceptuální návrh
- logický návrh
- fyzický návrh

1.5.1. Konceptuální návrh

Cílem konceptuálního návrhu je vytvoření ER modelu. Schéma konceptuálního návrhu má tyto kroky:

- identifikace entit – definujeme hlavní objekty databáze – entity (název, popis)
- identifikace relací – první návrh ER diagramu, nastíníme vztahy mezi entitami a konkretizujeme vztahy mezi nimi
- identifikace a spojení atributů s entitami nebo relacemi – určení názvu entity, jmen atributů, popis atributů, datové typy atributů, ...
- určení domén atributů – charakterizujeme množinu hodnot, z nichž čerpají hodnoty jeden nebo více atributů
- určení atributů, které budou primárními a kandidátními klíči
- specializace/generalizace (tento krok je volitelný) – modelování podtříd a nadtříd
- kontrola redundance – nejdříve přezkoumáme relace 1:1 (mohli jsme identifikovat dvě entity pro stejný objekt) a dále odstraníme redundance, tzn. takových relací, u kterých lze získat stejnou informaci prostřednictvím jiné relace
- kontrola podpory uživatelských rozhraní – popis transakcí nebo sledování cest transakcí
- posouzení konceptuálního návrhu s uživateli – diskuse s uživateli [1]

1.5.2. Logický návrh

Cílem logického návrhu je na základě vytvořeného ER diagramu navrhnout již konkrétní struktury tabulek a provést jejich kontrolu z hlediska normalizace a integrity. Logický návrh má tyto kroky:

- vytvoření tabulek – vytváříme tabulky dle ER modelu (název, seznam atributů, definice primárních a cizích klíčů – definice relací mezi tabulkami)
- kontrola tabulek s využitím normalizace – kontrolujeme, zda tabulky vytvořené v předchozím kroku splňují pravidla normalizace a integritních omezení. V případě problému provedeme rekonstrukci modelu dat nebo tabulky
- kontrola na podporu uživatelských transakcí – z hlediska ER modelu
- posouzení logického návrhu s uživateli – diskuse [10]

1.5.3. Fyzický návrh

Na úrovni fyzického návrhu již realizujeme v předchozích návrzích připravenou databázi. Realizace probíhá v těchto krocích:

- návrh a implementace tabulek (CREATE TABLE)
- návrh reprezentace odvozených položek (SELECT, INSERT)
- návrh zbývajících integritních omezení
- organizace souborů a indexů
- návrh uživatelských pohledů (CREATE VIEW)
- návrh bezpečnostních mechanismů
- kontrolované zavedení redundance
- vyladění systému [10]

1.6. SQL

SQL (Structured Query Language) je dnes již standardním jazykem, který používáme při komunikaci s relačními databázemi. Pro získání jakýchkoli informací z databáze používáme tzv *dotaz (Query)*, na jehož základě nám databáze poskytne požadovanou odpověď. SQL je nejrozšířenějším jazykem, který umožňuje tvořit databázové dotazy.

SQL není procedurálním jazykem jako jsou například C, Pascal, Basic, COBOL a mnohé další. Nicméně někteří dodavatelé již nabízí procedurální rozšíření jazyka SQL. Uveďme například Procedural Language/SQL společnosti Oracle nebo Transact-SQL v databázích firmy Microsoft. Ale tato rozšíření můžeme označit za nové jazyky, jejichž podmnožinou je jazyk SQL.

Můžeme tedy říct, že SQL slouží ke správě a údržbě relačních databází. Není vhodný k obecnému programování aplikací. [7]

1.6.1. Jazyk DDL (Data Definition Language)

DDL zahrnuje příkazy SQL, které slouží k tvorbě databázových objektů (např. tabulky, pohledy a indexy) a úpravě jejich struktury. Součástí jazyka DDL jsou příkazy, které obsahují klíčová slova CREATE, ALTER a DROP. Mají vliv na kontejnery, které data uchovávají, nikoli však na data samotná.

Příkazy jazyka DDL:

- CREATE – vytvoří nový databázový objekt, který je v příkazu uveden (CREATE DATABAZE, CREATE TABLE, CREATE INDEX a CREATE VIEW)
- ALTER – tímto příkazem můžeme měnit definice existujícího databázového objektu, který je v příkazu uveden (ALTER TABLE, ALTER DATABASE, ALTER SYSTEM, ALTER USER, ALTER SESSION atd.)
- DROP – tento příkaz odstraní nebo zruší existující databázový objekt, který je v příkazu uveden [7]

1.6.2. Jazyk DQL (Data Query Language)

Obsahuje takové příkazy SQL, které načítají data z databáze. Přestože se jedná o velmi důležitou součást jazyka SQL, je reprezentován pouze příkazy založených na klíčovém slově SELECT. [7]

1.6.3. Jazyk DML (Data Manipulation Language)

Tato skupina příkazů umožňuje uživatelům přidávat data do databáze v podobě řádků a tabulek. A také existující data odebírat či měnit. Zahrnujeme sem příkazy s klíčovými slovy INSERT, UPDATE a DELETE. [7]

1.6.4. Jazyk DCL (Data Control Language)

Tyto příkazy dovolují správcům řídit přístup k datům v databázi a používat různá systémová oprávnění SŘBD (sytém řízení báze dat). Příkladem může být funkce na vypnutí nebo zapnutí databáze. Do této skupiny patří příkazy založené na klíčových slovech GRANT a ALTER. [7]

1.6.5. Datové typy jazyka SQL

Nejmenší pojmenovanou jednotkou dat, na kterou se můžeme v relační databázi odkazovat, je sloupec. Každému takovému sloupci je nutné přiřadit jedinečný název a datový typ. *Datový typ* je kategorie, která určuje formát příslušného sloupce.

Datové typy nám přinášejí tyto výhody:

- omezují data ve sloupci na znaky, a to takové, které mají smysl z hlediska konkrétního datového typu
- poskytují uživatelům databázových systémů soubor užitečných vlastností (např. když sčítáme číslo s číslem, výsledkem bude opět číslo)
- Jsou důležitým pomocníkem relačního systému řízení báze dat (RSŘBD) při efektivním ukládání dat sloupců.

Jazyk SQL podporuje tři kategorie datových typů. *Předem definované datové typy* jsou k dispozici jako nativní součást RSŘBD. *Složené datové typy* uchovávají pole nebo kolekce hodnot předem definovaných datových typů. *Uživatelsky definované datové typy* umožňují uživatelům databáze definovat vlastní datové typy (pouze v případě že je to podporováno RSŘBD).

Protože návrh a poté i samotnou databázi budu tvořit v databázovém systému Microsoft SQL Server, uvedu zde podporované datové typy tohoto systému:

- BIGINT – celá čísla od -2^{63} do $2^{63} – 1$
- CHARACTER (CHAR) – řetězce s pevnou délkou až do 8000 znaků
- DECIMAL – pevná přesnost a rozsah od $-10^{38} + 1$ do $10^{38} – 1$
- FLOAT – čísla s plovoucí desetinnou čárkou od $-1{,}79 \times 10^{308}$ do $-2{,}23 \times 10^{-308}$, 0, nebo v rozsahu $2{,}23 \times 10^{-308}$ do $1{,}79 \times 10^{308}$
- INTEGER – celá čísla od -2^{31} do $2^{31} – 1$
- NCHAR – řetězce v kódování Unicode s pevnou délkou až do 4000 znaků
- NUMERIC – viz typ DECIMAL
- NVARCHAR – řetězce v kódování Unicode s proměnnou délkou až do 4000 znaků
- REAL – čísla s plovoucí desetinnou čárkou v rozsahu od $-3{,}40 \times 10^{38}$ do $-1{,}18 \times 10^{-38}$, 0, $1{,}18 \times 10^{-38}$ do $3{,}40 \times 10^{38}$
- SMALLINT – celá čísla od -32 768 do 32 767
- VARCHAR – řetězce s proměnnou délkou až do 8000 znaků

Microsoft SQL Server nabízí ještě tato rozšíření standardních datových typů:

- BINARY – binární data s pevnou délkou až do 8000 bajtů
- BIT – celočíselná data s hodnotou buď 1 nebo 0
- DATETIME – datum a čas od 1. ledna 1753 do 31. prosince 9999
- IMAGE – binární data s proměnnou délkou
- MONEY – finanční datové hodnoty od -2^{63} do $2^{63} – 1$
- NTEXT – data v kódování Unicode s proměnnou délkou
- SMALLDATETIME – datum a čas od 1. ledna 1900 do 31. prosince 2079
- SMALLMONEY – finanční datové hodnoty od -214 748,3648 do 214 748,3647
- TEXT – řetězec s proměnnou délkou
- TIMESTAMP – číslo jedinečné v rámci databáze, které je aktualizováno při každé změně řádku
- TINYINT – celá čísla od 0 do 255
- UNIQUEIDENTIFIER – globálně jedinečný identifikátor
- VARBINARY – binární data s proměnnou délkou až do 8000 bajtů [7]

1.6.6. Transakce

Databázová transakce je soubor příkazů, který je databázovým uživatelem koncipován tak, aby byl zpracován jako nedělitelná jednotka. Z kontextu je tedy zřejmé, že transakce musí být kompletně úspěšná nebo kompletně neúspěšná. Navíc příkazy řídící databázové transakce nejsou přesně shodné se syntaxí příkazů jazyka SQL, ale zásadně ovlivňují chování těch příkazů jazyka SQL, které jsou součástí transakce. [7]

1.6.7. Pohledy

Pohledy poskytují uživatelům databáze spoustu výhod, protože umožňují přizpůsobit data individuálním podmínkám a požadavkům. Navíc zobrazují data přehledněji. Správně vytvořené pohledy příliš nezvyšují režii a nevyžadují ukládání dat. Pohled je v podstatě uložený dotaz SQL, na který se můžete odkazovat stejně, jako by se jednalo o skutečnou tabulku. Fyzicky ale tyto tabulky neexistují. [7]

1.6.8. Triggers – spouště

Spouští označujeme uloženou proceduru, jež se automaticky vykoná v případě určité, předtím definované události, která nastává při manipulaci s údaji. Například při vkládání nebo vymazávání údajů v tabulce a podobně. Klíčovou vlastností spouští je, že se nespouští přímo, ale jsou navázány na příkazy, které modifikují údaje, jako INSERT, DELETE, UPDATE. Používají se ke kontrole zadávaných údajů, zajištění datové integrity a podobně. Spoušť je možné aktivovat před, po a nebo místo příkazu, který ji aktivoval. [4]

1.7. Webové aplikace

V této kapitole se jen velmi stručně zmíním o možné aplikační nadstavbě databázového systému, nebo spíše jeho využití ve webových aplikacích. Toto téma není předmětem této práce, proto se bude jednat jen o velmi stručné seznámení a nástinu možné další cesty k zpříjemnění a zjednodušení práce s databázovým systémem pro uživatal, který není do problematiky zasvěcen.

1.7.1. HTML a CSS

HTML neboli *HyperText Markup Language* je skriptovací jazyk pro hypertext. Je jedním z jazyků, pravděpodobně nejrozšířenějším, pro vytváření stránek v prostředí World Wide Web. Je charakterizován množinou značek, tzv. *tagů*, a jejich atributů definovaných pro danou verzi. Popisuje grafickou podobu stránky v prohlížeči. Má i svoji syntaxi, kterou je nutné dodržovat, nicméně zároveň je jazyk HTML velmi přizpůsobivý.

Pro popis způsobu zobrazení stránek napsaných v jazycích HTML využíváme tzv. *kaskádové styly* (Cascading Style Sheets) se zkratkou CSS. Hlavním smyslem CSS je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. [8]

1.7.2. PHP

PHP (Personal Home Page) je skriptovací programovací jazyk, který je určený zejména pro programování dynamických internetových stránek. Nejčastěji bývá vkládán přímo do struktury jazyka HTML, což můžeme využít při tvorbě webových aplikací. Díky své jednoduchosti použití, bohaté zásobě funkcí, a tomu, že kombinuje vlastnosti více programovacích jazyků, což nechává vývojáři částečnou svobodu, velmi oblíbeným jazykem. [8]

1.7.3. Tvorba aplikace

Nejprve je nutné projekt velmi důsledně a pečlivě a především komplexně připravit. Provést analýzu trhu, promyslet ekonomickou stránku projektu, pečlivě zvážit, jestli jsem schopen ho dotáhnout do konce. A pokud ano, tak jestli ho budu vyvíjet sám nebo v týmu.

Před samotným zahájením projektu je nutné zpracovat detailní analýzu., aby projekt nezhavaroval za běhu a nedošlo ke ztrátám.

Dalším krokem je modelace databázového systému a jeho struktury. Je vhodné to udělat před tím, než začneme programovat přímo aplikaci, protože pokud v průběhu tvorby aplikace budeme měnit databázovou strukturu, mohou nám přestat fungovat části kódů.

Následně se můžeme pustit do tvorby samotné aplikace. Můžeme začít psát aplikace jak strukturálně, tak i objektově, protože i objektový model je v PHP 6 velmi dobře zpracován.

Nakonec v poslední fázi tvorby aplikace všechno důsledně testujeme a ladíme. [8]

2. Analýza současného stavu

2.1. Představení společnosti

Společnost, se kterou vás nyní seznámím, a kterou budu analyzovat, sídlí v Brně na Mojmírově náměstí 2 a kancelář se nachází na adrese Křížíkova 70 rovněž v Brně. Jmenuje se Ready Control s.r.o. a její IČ je 29364728 a DIČ je CZ29364728.

Obrázek 12: *Logo společnosti*
Zdroj: www.readycontrol.cz

Ready Control s.r.o. je česká společnost, která zajišťuje svým klientům kompletní servis od společnosti RWE. Poskytuje poradenství v oblasti energetiky se zaměřením na optimalizaci nákladů spotřebitele.

Zprostředkovává tedy prodej elektřiny a zemního plynu. Trh s energiemi rychle roste a s ním právě i zmiňovaná společnost Ready Control s.r.o., která v současnosti dosahuje obratu cca 1 000 000 Kč za měsíc, a zaměstnává 30 zaměstnanců, a její tým se neustále rozrůstá. Podnikání v tomto odvětví v České republice umožnila takzvaná liberalizace trhu energií (otevření trhu s elektrickou energií a plynem). Tím se elektřina či plyn staly standardním zbožím. Alternativní dodavatelé je nakupují od výrobců za lepších cenových podmínek – získají totiž množstevní slevu – a dále je pak mohou prodávat za podmínek, které si alternativní dodavatel určí, dohodne s klientem a přizpůsobí tržnímu prostředí. Dnes už je distributor (ten, kdo energii přivádí) vždy odlišný od dodavatele (ten, kdo energii prodává) a všechny části technického zajištění dodávky energie do domu jsou majetkem příslušné distribuční společnosti, která se nemění ani se změnou dodavatele (ČEZ Distribuce a.s., E.ON Distribuce a.s., PRE Distribuce a.s., RWE GasNet, s.r.o., Pražská plynárenská distribuce a.s., a další). A proto se nemění ani technické zajištění dodávek energie.

2.2. Historie firmy

Společnost byla zapsána do obchodního rejstříku pod obchodní firmou Ready Control s.r.o. 27. července 2012. V prvním roce své existence společnost zprostředkovávala prodej pro malé alternativní dodavatele. V této fázi měla firma dvoučlenný management, jedna osoba zastávala post v back office, která se starala o administrativu a péči o zákazníka. Administrativa zahrnovala evidenci smluv, ať už zaměstnaneckých, tak i smluv se zákazníky a dalšími dodavateli, a další. Finanční služby v účetnictví a IT byly najímány externě. Firmě se dařilo a v roce 2013 vyhrála výběrové řízení pro RWE. Prudce vzrostl počet klientů, a tedy evidovaných smluv, a vedlo to i ke vzniku nových pracovních pozic – a to pozice v call centru, účetní a rozrostl se obchodní tým. V oblasti IT společnost stále využívá externisty. Nyní se již společnost stala dodavatelem elektrické energie a zemního plynu. Klade důraz na péči o zákazníka. Kvůli tomu vytvořil management nový produkt – POSEL, který zahrnuje ochranu a péči o zákazníka, a to i po uzavření smlouvy.

Swot analýza:

Silné stránky

- velké klientské portfolio
- produkt POSEL
- kvalitně proškolení zaměstnanci
- trvalá péče o zákazníka

Slabé stránky

- informační oddělení
- zpracování a evidence smluv v programu MS Excel

Příležitosti

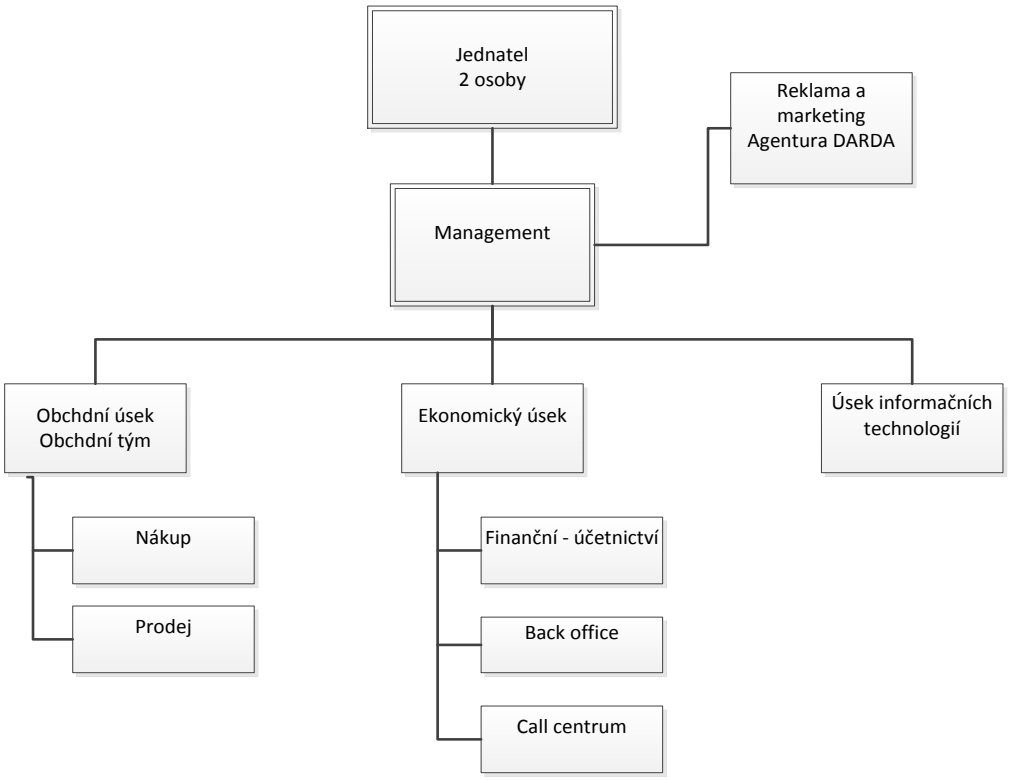
- oslovení nových klientů
- vytvoření nových produktů

Hrozby

- příliš proměnlivý tržní prostředí

- legislativa

2.3. Organizační struktura



Obrázek 13: *Schéma organizace společnosti*
Zdroj: Vlastní

2.4. Informační technologie

Hardware:

Ve společnosti se používají jak osobní počítače, tak notebooky. Na hardware nejsou až na výjimky kladeny příliš velké nároky. Především musí zvládnout kancelářské aplikace a online komunikaci.

V kancelářích používají sedm stolních počítačů *ALFA Basic Plus W8*, na kterých je nainstalován operační systém Microsoft Windows XP s těmito parametry:

- CPU Intel Pentium G2030 3,0 GHz

- RAM 4 GB DDR3
- VGA Intel HD
- HDD Seagate 500GB

Dále používají osm notebooků *Acer Aspire E13* se systémem Microsoft Windows 7 s těmito parametry:

- CPU Intel Celeron N2840 2,5 GHz
- RAM 4 GB DDR3
- HDD 500 GB
- int. VGA, displej 1366x768 matný
- wlan, glan, bt, ...

dvě stanice PC ALL IN ONE *Acer Aspire ZC-605* se systémem Microsoft Windows 8 s těmito parametry:

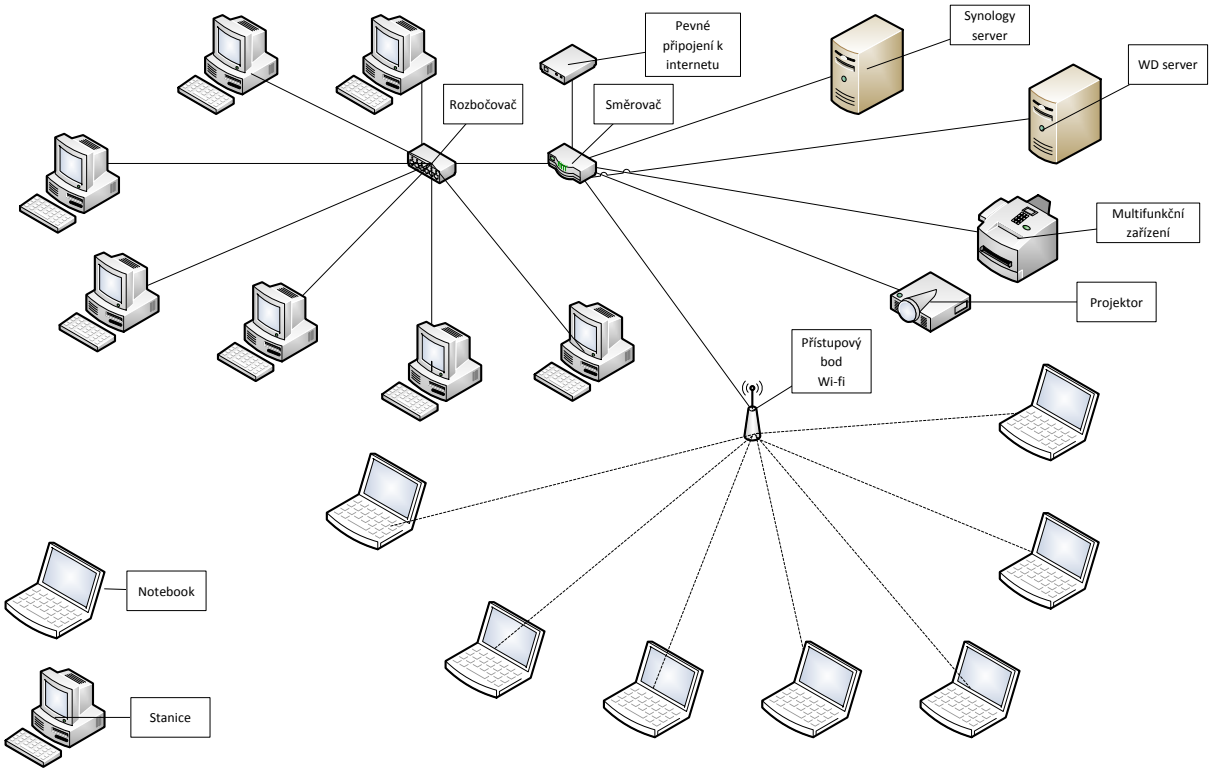
- CPU Intel Celeron Dual – Core 1017U 1,6 GHz
- RAM 2 GB DDR3
- WXGA + HD 1600x900
- HDD 500GB SATA
- USB kbd + mouse, wi-fi, ...

a nakonec dva NAS serverové počítače. *Western Digital My Cloud EX2* s parametry:

- NAS – datové úložiště k místní síti
- CPU 1,2 GHz
- RAM 512 MB DDR3
- 1x2TB HDD + ještě jeden volný slot na další HDD

a *Synology DS115j* s parametry:

- NAS – datové úložiště k místní síti
- CPU Marvel Armada 800 MHz
- RAM 256 MB DDR3
- HDD 2TB (maximální kapacita vnitřního úložiště 6TB)



Obrázek 14: *Síťová struktura společnosti*
Zdroj: Vlastní

Software:

V současné době ve společnosti využívá finanční oddělení účetní systém POHODA. Pro správu klientů, jejich plateb a podobně používá back office a management podnikový informační systém INEX. Pro šifrování a certifikace používají systém CRYPTA. Dále společnost vlastní deset licencí na Microsoft Office 2013, 7 licencí na Open Office 4.1.1 a 17 licencí antivirového programu AVG 2015.

2.5. Zpracování dat

Firma používá několik informačních systémů, a každý si ukládá vlastní data do své vlastní databáze. Ty jsou dále zálohovány na firemních diskových serverech. Data ovšem nejsou konzistentní, a proto by bylo vhodnější vytvořit na diskových serverech datový sklad, ve kterém by byla data konzistentní, dostupná a snadno zpracovatelná. Mě zajímá především databáze evidence obchodních smluv uzavřených konkrétními zaměstnanci, které si firma eviduje v programu Microsoft Excel 2013.

2.6. Vymezení aktuálních problémů

Firma Ready Control s.r.o. používá několik informačních systémů (Pohoda, INEX) a datová úložiště NAS. Bohužel je však datový model decentralizovaný a nekonzistentní. Největší problém představuje databáze evidence uzavřených obchodních smluv. Je vedena, jak jsem již v článku 2.7 zmínil, v programu Microsoft Excel 2013. To s sebou nese mnohá úskalí. Největším problémem je kontrola jedinečnosti každé smlouvy, protože někteří zaměstnanci, ať už vědomě nebo nevědomě, občas předloží vkladateli dat do „databáze“ již uzavřenou a vloženou smlouvu, za kterou byla provize již zaplacena, a při opětovném vložení by byla provize proplacena znovu. Aktualizace databáze je komplikovaná, protože s ní pracuje jak vedení, tak další administrativní pracovníci. Z čehož vyplývají další problémy jako úroveň zabezpečení, která je v podstatě nulová. Dále nepřehlednost, rozmístění dat do několika různých souborů a jejich nenormalizované ukládání. A v neposlední řadě trpí tím, že ji není možné reálně aktualizovat tak, aby všichni uživatelé měli přístup ke stejným aktuálním datům současně. Mým úkolem bude vytvořit databázi, která již nebude těmito nedostatky trpět.

Hlavním cílem mé bakalářské práce je tedy návrh a vytvoření datového modelu, jehož stěžejním úkolem bude evidence a správa obchodních smluv uzavřených konkrétním zaměstnancem.

V současné době pracuje ve firmě cca 30 obchodních zástupců, kteří měsíčně připraví a realizují v průměru více jak sto smluv za jeden kalendářní měsíc. Z pohledu evidence jde o tři neustále se opakující procesy, a to vznik nové smlouvy, změna či úprava stávající smlouvy a nakonec ukončení již existující smlouvy.

Současný proces evidování smluv

Obchodní zástupce po úspěšném jednání s klientem podepisují listinné vyhotovení smlouvy se všemi náležitostmi. Takovou smlouvu obchodník přiveze na firmu. Buď ji hned odevzdá na back office nebo si ji nechá v šanonu u sebe a odevzdá ji hromadně se všemi, které v kalendářním měsíci realizuje. Ale to je spíš výjimka, kdy obchodník předpokládá realizaci více smluv v jednom či několika málo po sobě jdoucích dnech. V dalším kroku tedy se smlouvami pracuje úředník v back office. Bere jednu smlouvu po druhé, ověřuje jejich správnost a v případě, že je vše v pořádku, zadává údaje ze

smluv do tabulky vytvořené v programu Microsoft Excel 2013. A nyní se dostáváme k úskalím současného systému evidence.

V první řadě budeme uvažovat evidenci v jednom kalendářním měsíci. Tady je největším problémem již zmiňovaná nekonzistence dat. Úředník zaeviduje veškeré smlouvy, které mu předloží obchodní zástupci. Soubor s daty následně úředník pošle managementu, který sleduje výslednost zaměstnanců, a potažmo výdělek firmy v daném měsíci. Úředník data v rámci jednoho měsíce doplňuje data stále do stejného souboru a posílá jej managementu pod stejným názvem, což je provedení komplikace v tom, že si musí ověřovat, zda otvírá aktuální soubor. A pokud je to aktuální soubor, musí zjišťovat kolik smluv v něm je nových.

Dále je velkým problémem, že pro každý měsíc existuje samostatný soubor s evidencí smluv, což generuje další problémy a komplikace. Pokud chcete zjistit, jestli daná smlouva již existuje, musíte procházet soubor po souboru a hledat buď po jednotlivých záznamech, nebo pomocí filtru, ale bohužel jen v rámci jednoho souboru. Za dobu existence jsou to již tisíce smluv a desítky souborů. A stává se, že obchodní zástupce předloží stejnou smlouvu opakovaně, a pak je mu provize proplacena dvakrát. Na tuto praktiku jsem byl upozorněn a je hlavním důvodem, proč firma chce změnit způsob a formu evidování smluv.

Komplikované je, i když chcete smlouvu zrušit či upravit. Musíte opět hledat konkrétní smlouvu soubor po souboru. Zákazníci do výpovědi sice uvedou číslo smlouvy, ale již ne datum jejího uzavření. Totéž platí při úpravě smlouvy. Opět se smlouva identifikuje přes její číslo, což nás vede zase k hledání soubor po souboru. Na to navazuje aktualizace souborů. Managementu se musí posílat opětovně nově aktualizované soubory, který musí následně mazat staré verze souborů. Tady hrozí i riziko ztráty dat.

Při tomto způsobu je problematické i zálohování dat, protože musíte aktualizovat i zálohované soubory, jinak byste se v nich za chvíli nevyznaly. Nebo vytvoříte velké množství záloh se stejným důsledkem. A dalším rizikem je lidský faktor, který nelze úplně odstranit, ale je možné zabránit nechtěným ztrátám dat a další lidské chyby (chyby při zadávání dat, například špatně zadané datum apd.) ošetřit. V následující kapitole uvedu vlastní řešení evidence smluv, které odstraní všechny výše zmíněné problémy současného systému evidování smluv.

3. Vlastní návrh řešení

3.1. Výběr databázového systému

Databázový systém jsem vybíral z těchto možností:

- MySQL
- Oracle
- PostgreSQL
- MSSQL

Vzhledem k tomu, že veškerý software firmy běží na platformě Microsoft Windows, a který je pro zaměstnance v podstatě jediným známým uživatelským rozhraním, vybral jsem nakonec právě databázový systém Microsoft SQL Server 2012, jehož uživatelské rozhraní bude zaměstnancům firmy nejbližší a nejsnáze se s ním seznámí. Navíc vzhledem k rozsahu databáze a ceně systému je dle mého názoru tato volba ideální. Nyní si systém MSSQL server v krátkosti představíme.

3.1.1. Microsoft SQL Server 2012 Standard edition

Tato edice je primárně určená k provozu aplikací firemních oddělení, a to nenjen databázových, ale i Business Inteligence aplikací pro menší firmy a organizace nebo jejich oddělení. Některé funkce typicky směrované do prostředí velkých firem Standard edition nezahrnuje. Podporuje 16 procesorových jader, 64 GB RAM, jeden virtuální stroj a dva uzly pro *failover clustering*. Zahrnuje i některé pokročilejší funkce jako *Policy – based management*, podporu technologie *Spatial* či multidimenzionální sémantický Business Inteligence model.

Zde uvádím některé typické scénáře pro nasazení:

- aplikace firemních oddělení požadujících dobrou spravovatelnost a jednoduché použití
- aplikace pro online zpracování transakcí (OLTP) v malém až středním objemu
- systémy pro podporu rozhodování, například v sektoru SMB nebo v autonomních pobočkách požadujících základní funkce pro generování sestav a analýzy [4]

3.1.2. Microsoft SQL Server 2012 Enterprise edition

Tato platforma je ucelená a splňuje vysoké nároky podnikových aplikací, jak pro online zpracování transakcí v datových centrech nebo pro datové sklady. Umožňuje také konsolidaci serverů a realizaci online zpracování velkého objemu transakcí a generování sestav. Rovněž zajišťuje obchodní kontinuitu a zkracuje čas potřebný k obnovení po havárii, což umožňují funkce chránící data před nákladnými lidskými chybami. Můžeme jej nasadit v rámci privátního cloudu a ve velkých centralizovaných Business Intelligence řešeních. Umí vytvořit infrastrukturu s ověřenými schopnostmi zpracování velkých množství dat a vysokého podnikového zatížení. Samozřejmostí je splnění požadavků na ochranu osobních dat a soulad s legislativními normami a nabízí integrované funkce pro ochranu dat před neoprávněným přístupem. Verze Enterprise nabízí správu infrastruktury s automatickou diagnostikou, optimalizací a konfigurací s cílem snížit provozní náklady a zároveň omezit nutnost údržby a správy velkých objemů dat. Umožňuje dotazování a analýzu velkých množství dat v datových skladech a datových tržištích a ulehčuje tak získávání širšího pohledu na tato data. Typické scénáře pro nasazení edice Enterprise:

- provoz nenahraditelných aplikací pro správu dat se škálovatelností, vysokou dostupností a zabezpečením na podnikové úrovni
- správa online zpracování transakcí (OLTP) ve velkém objemu
- pokročilá analýza velkých objemů dat v datových skladech
- generování sestav na základě analýzy velkých objemů dat [4]

3.1.3. Microsoft SQL Server 2012 Business Intelligence edition

Edice Business Intelligence je novinkou ve verzi 2012 a asi nejlépe podtrhuje význam Business Intelligence a všeho, co tento pojem obsahuje, tedy analýz, multidimenzionálních databází a dolování dat v podnikové praxi. Nabízí firmám a organizacím kompletní sadu škálovatelných Business Intelligence funkcí včetně *Power View* a *Power Pivot*. Je přizpůsobena koncepci samoobslužné Business Inetlligence pro firmy, které nevyžadují výkonné online transakční zpracování (OLTP). [4]

3.1.4. Microsoft SQL Server 2012 Specializované edice

No a podobně jako u předchozích verzí jsou kromě hlavních k dispozici i některé účelově specializované edice. [4]

3.1.4.1. Developer edition

Jak už je z názvu patrné, jedná se o edici určenou pro vývojáře. Obsahuje všechny vlastnosti a funkce edice Enterprise, ale licenčním ujednáním je směrovaná výhradně pro účely vývoje, testování a předvádění. [4]

3.1.4.2. Web edition

Tato edice nabízí nízké náklady, vysokou škálovatelnost pro vysoko dostupné webové aplikace či hostovaná řešení a vysokou dostupnost internetových prostředí webových služeb. Typický scénář nasazení představují sdílená a dedikovaná hostitelská řešení. [4]

3.1.4.3. Express edition

Edice Express je k dispozici bezplatně a je ideální pro studium a vytváření aplikací pro klientské počítače a malé servery a pro distribuci nezávislými výrobci softwaru. Edice je limitovaná na využití jednoho procesoru, 1 GB RAM a velikosti databáze 10 GB. Typické scénáře pro nasazení edice Express:

- základní a studijní databáze
- funkčně bohaté aplikace pro klientské počítače
- bezplatná práva na distribuci pro nezávislé výrobce softwaru (ISV)

Edice může také sloužit studentům nebo vývojářům „hobby“ aplikací pro vývoj a zavádění jejich aplikací. Rovněž je zdarma pro nezávislé dodavatele softwaru a výrobce hardwaru, kteří ji mohou distribuovat nebo zabudovat do svých aplikací a produktů. Express edition je v rozsahu základní funkcionality plně kompatibilní se všemi edicemi SQL Serveru. [4]

Ze všech výše zmíněných edicí Microsoft SQL Serveru 2012 jsem vybral edici *Standard edition*. Svým rozsahem plně postačuje pro potřeby firmy Ready Control s.r.o.. Rovněž tak cena je odpovídající pro rozsah využití této edice. Navíc do budoucna je možné a pravděpodobné mnohem komplexnější využití tohoto databázového systému a ve větším rozsahu.

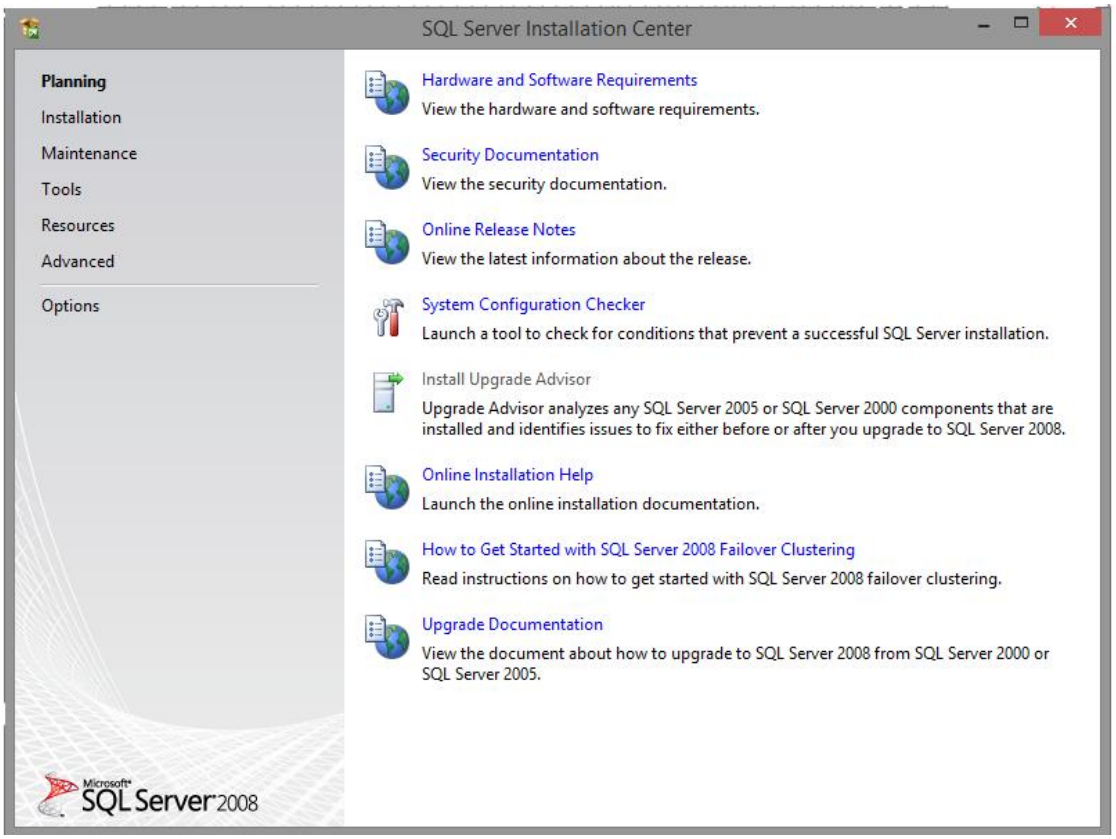
3.2. Instalace Microsoft SQL Server 2012 Standard edition

V případě firmy Ready Control s.r.o. jsem se rozhodl pro instalaci MSSQL Serveru 2012 do virtuálního prostředí. Budeme tedy používat virtuální server hostovaný na jednom z jejich fyzických serverů. Virtuální server si můžeme představit jako aplikaci spuštěnou na jiném počítači nebo serveru. Tomuto serveru říkáme hostitelský (primární) server a jeho operační systém je primárním operačním systémem. Na hostitelském serveru se disk virtuálního serveru bude jevit zpravidla jako jeden soubor, bude dokonale separovaný, jako by běžel na samostatném hardwaru. Virtuální servery jsou navzájem nezávislé, proto pád jednoho neovlivní pád jiného. Navíc umožňují jednoduchou a rychlou obnovu po pádu systému, protože jde o fyzický soubor s obsahem jeho pevných disků. Což dále umožňuje tento soubor snadno zálohovat a archivovat a v případě potřeby je i z této zálohy obnovit.

Databázové prostředí SQL serveru je komplexní softwarové prostředí, které musíme nainstalovat a nakonfigurovat. K tomu použijeme nástroj Microsoft SQL Serveru 2012, který se jmenuje *SQL Server Installation Center*. Po spuštění se otevře okno rozdělené na dvě poloviny, kde levá polovina obsahuje názvy karet:

- Planning
- Installation
- Maintenance
- Tools
- Resources
- Advanced
- Options

Na pravé polovině vidíme možnosti a nástroje, které můžeme použít. Pomocí tohoto nástroje nainstalujeme a nakonfigurujeme kompletní Standardní edici MSSSQL 2012 jako virtuální server. Vedení firmy plánuje do budoucna širší využití včetně Business Inteligence. Požadavky instalace na hardware 64bitové architektury serverů jsou procesor AMD Opteron, AMD Athlon 64, Intel Xeon s podporou Intel EM64T, případně Intel Pentium IV s podporou EM64T. Taktování procesoru by mělo být minimálně 1,4 GHz a minimum operační paměti RAM je 512 MB, ale praktické doporučené minimum je 2 GB. Pomo



Obrázek 15: *Instalační rozhraní nástroje SQL Server Installation Center*
Zdroj: Vlastní

Při instalaci jsem prostřednictvím *SSIC* provedl tyhle kroky:

- identifikace problémů instalace (proběhla automaticky a bez problému)
- výběr komponent pro instalaci (kompletní edice Standard)
- konfigurace instalování instance (defaultní)
- konfigurace přístupových účtů (kombinovaná autentifikace – jak z OS Windows, tak přímo přes SQL Server, zatím pouze účet administrátora serveru)

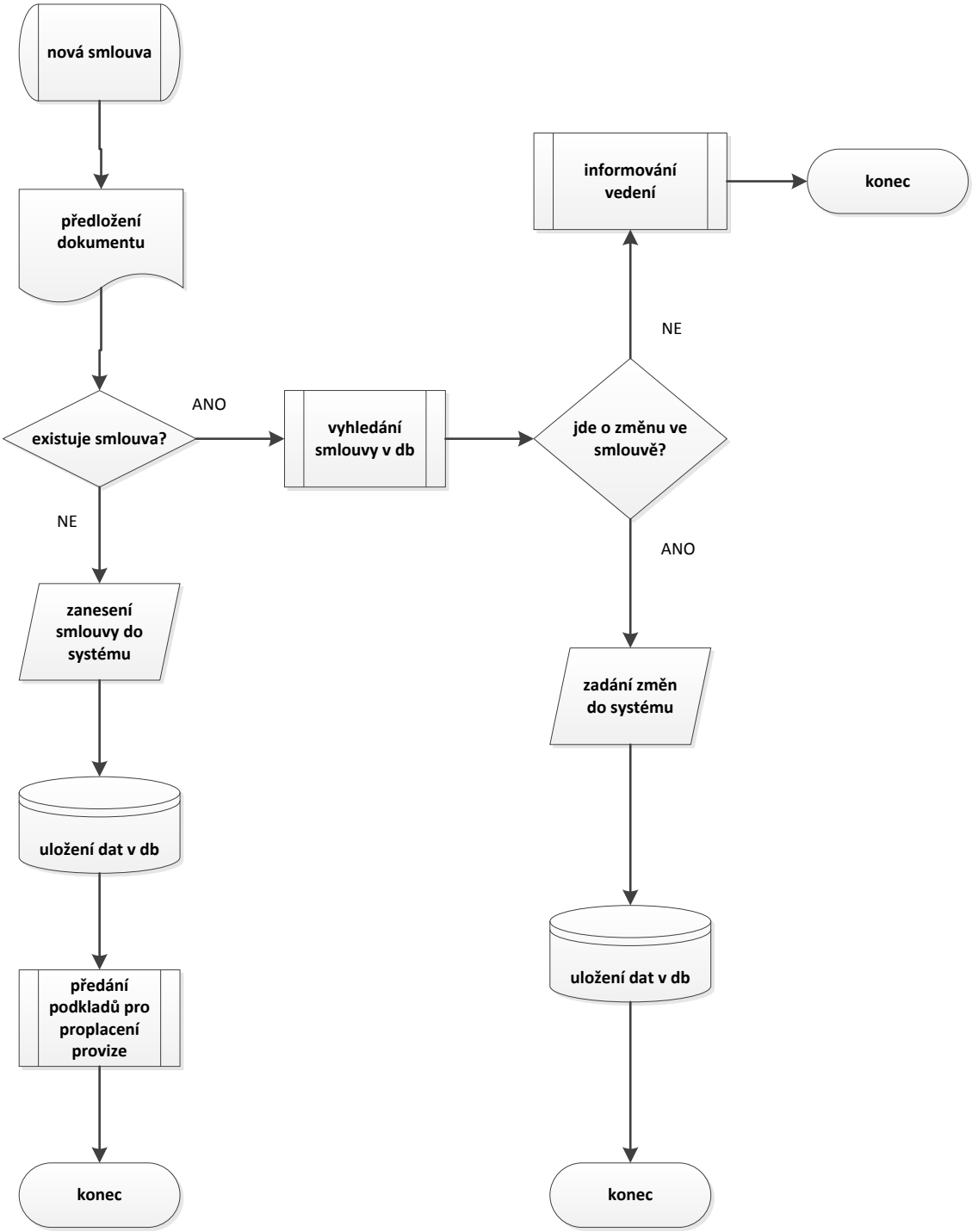
- konfigurace analytických služeb (multidimensional and data mining)
- konfigurace reportovacích služeb
- závěrečná kontrola konfigurace a vlastní instalace [4]

3.3. Definice procesů a požadavků databáze

Celou evidenci smluv z hlediska zaznamenávání dat do databáze můžeme rozdělit do tří základních procesů. Prvním procesem je zaevidování nově uzavřené smlouvy, které končí i provizí pro obchodníka nebo prodejce, chcete-li. Druhým procesem, který vyplývá z nově zavedené služby společnosti, která se jmenuje *POSEL*. V režimu této služby jde o péči o klienta i po uzavření smlouvy, takže dochází k revizím a úpravám smluv na základě vývoje trhu a požadavků klienta. Druhým procesem je tedy revize a úprava smluv a jejich následné zaevidování do databáze. Třetím a posledním procesem je zrušení nebo zánik smlouvy a tím i smluvního vztahu. Zaniklou nebo zrušenou smlouvu následně ukončíme a v databázi označíme za již neplatnou.

3.3.1. Nová smlouva

Z hlediska evidence smlouvy v databázi není důležitý proces samotného obchodního jednání a vzniku smlouvy. Podstatný je uzavření smluvního vztahu a jeho písemné stvrzení. Smlouvu v papírovém provedení odevzdá obchodní zástupce úředníkovi v back office, který ji následně zapíše do systému. Systém prověří, zda se jedná skutečně o smlouvu novou, nebo zda jde pouze o revizi již existující smlouvy. A nebo, zda obchodní zástupce nepředkládá již existující smlouvu opakovaně. V případě potvrzení faktu, že se jedná o novou smlouvu, se informace uloží do databáze a papírová verze se odešle ke schválení managementu, který dále přesune na finanční úsek podklady pro vyplacení provize.

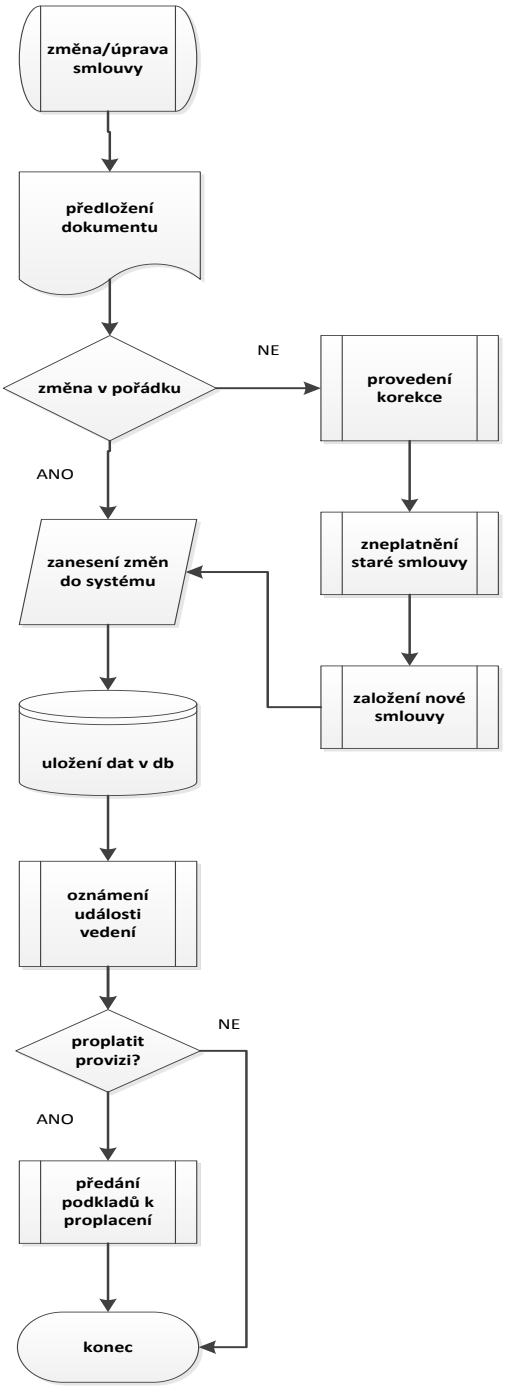


Obrázek 16: *Vývojový diagram – nová smlouva*
Zdroj: Vlastní

3.3.2. Úprava smlouvy

V dalším procesu si popíšeme proces, kdy obchodní zástupce předloží úředníkovi v back office smlouvu, která byla upravena či pozměněna na základě požadavku zákazníka nebo v důsledku změny vývoje na trhu. Úředník zneplatní starou verzi

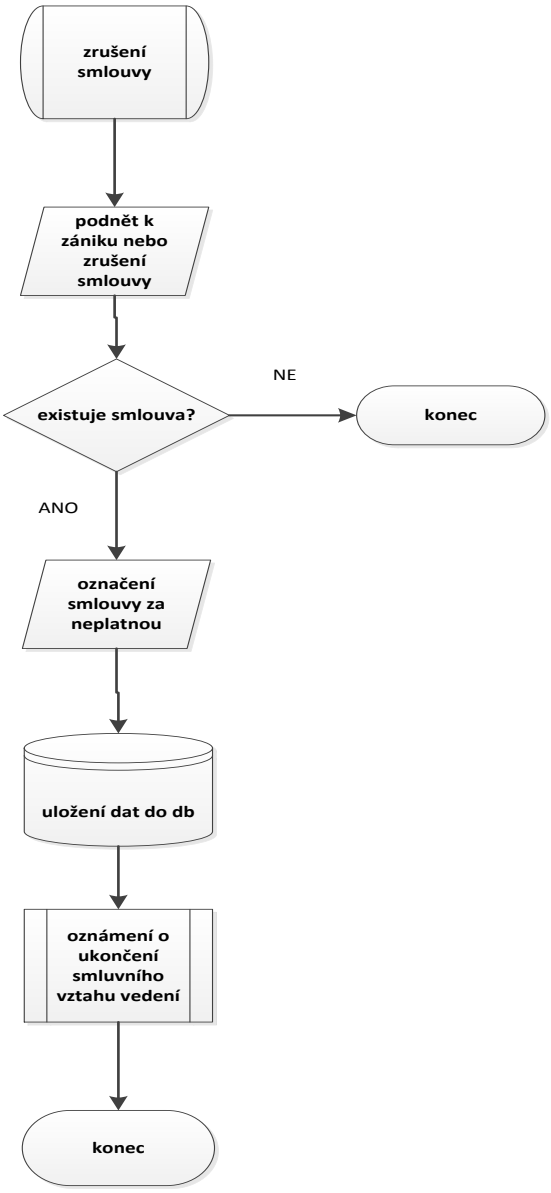
smlouvy a zapíše do systému smlouvu pod novým číslem. Provedené kroky uloží do databáze a odešle managementu ke schválení a přezkoumání, zda vzniká nárok na proplacení provize. V případě, že ano, odesílá potřebné podklady pro vyplacení provize na finanční úsek.



Obrázek 17: *Vývojový diagram – úprava smlouvy*
Zdroj: Vlastní

3.3.3. Zánik smlouvy a smluvního vztahu

Posledním procesem z hlediska evidence smlouvy v databázi je zánik smluvního vztahu a tím i smlouvy, prostřednictvím které je tento smluvní vztah uzavřen. Může k němu dojít jak ze strany klienta – výpověď smlouvy, tak ze strany prodejce – výpověď smlouvy. Vše na základě smluvních ujednání uzavřených v den podpisu konkrétní smlouvy. Na základě podkladů pro výpověď smlouvy se v systému zneplatní, ale zůstává dále archivována, i když jako neplatná.

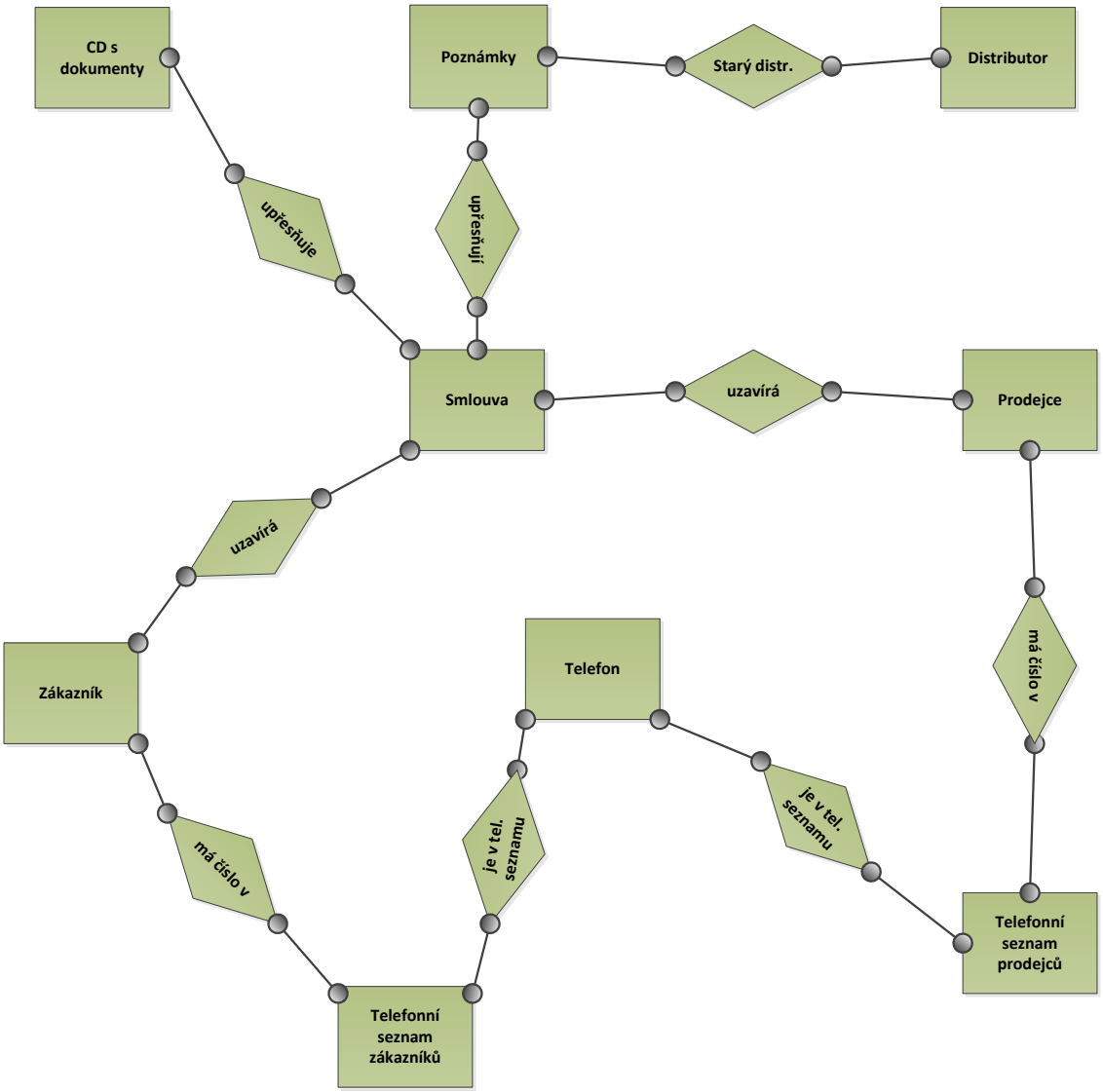


Obrázek 18: *Vývojový diagram – zánik smlouvy*
Zdroj: Vlastní

3.4. Konceptuální a logický návrh databáze

3.4.1. Konceptuální návrh databáze

Cílem této části práce je vytvořit entito – relační diagram databázového modelu. Tento model musí splňovat veškeré požadavky na databázi a co nejmenší redundanci dat. Budu postupovat od entit relací a atributů přes určení kandidátních klíčů až ke kontrole redundance dat a posouzení návrhu databázového modelu jako celku.



Obrázek 19: *Entito – relační diagram databáze*
Zdroj: Vlastní

3.4.2. Logický návrh databáze

Na základě vytvořeného ER digramu z předchozí fáze si připravíme již konkrétní tabulky s atributy, jejich primárními i cizími klíči a jejich datovými typy.

V tabulce č.1 *Distributoři* jsou uvedeni distributoři energií, kteří pokrývají trh v České republice. Navíc tato tabulka slouží jako číselník pro nadřazenou tabulku *Poznámky*.

Tabulka 1	**Distributor**
id distributor - pk	integer not null
Distributor	varchar (10) null

Tabulka 1: *Distributor*
Zdroj: Vlastní

Tabulka *Poznámky* obsahuje obecně informace o staré smlouvě klienta, jejím vypovězení a počátku dodávek od nového dodavatele – RWE nebo přímo Ready Control (Posel). S tabulkou *Distributor* je svázána přes cizí klíč *id_distributor*, který je zároveň primárním klíčem v tabulce *Distributor.*

Tabulka 2	**Poznámky**
id poznamky – pk	integer not null
Faktura	char (1) not null
Termin zacatku dodavek	date null
Vypovedni lhuta ve dnech	integer null
Stav	char (11) null
id distributor – ck	integer null

Tabulka 2: *Poznámky*
Zdroj: Vlastní

Ve třetí tabulce *CD Dokumenty* jsou v podstatě obdobné informace jako v tabulce č.2, ale jsou obsaženy v přiloženém CD, které je i se smlouvou odesíláno RWE.

Tabulka 3	**CD Dokumenty**
id dokumenty – pk	integer not null
Sipo	char (1) null
Faktura	char (1) null
Stara_smlouva	char (1) null

Tabulka 3: *CD Dokumenty*
Zdroj: Vlastní

Tabulka *Prodejce* obsahuje údaje o obchodníkovi, který smlouvu uzavřel či zpracoval. Je relačně propojena s tabulkou *Telefonní seznam prodejce* přes cizí klíč *id_tel_seznam_prod*. A dále ještě s tabulkou *Smlouvy* přes cizí klíč *id_smlouvy*.

Tabulka 4	**Prodejce**
id prodejce – pk	char (10) not null
Jmeno	varchar (15) not null
Prijmeni	varchar (30) not null
Ulice	varchar (40) null
cislo popisne	integer not null
Mesto	varchar (40) not null
PSC	integer not null
id smlouvy – ck	integer not null

Tabulka 4: *Prodejce*
Zdroj: Vlastní

Tabulka *Zákazník* obsahuje údaje o klientovi, který smlouvu s prodejcem uzavřel. I tato tabulka je relačně spojena s tabulkou *smlouvy* přes cizí klíč *id_smlouvy* a s tabulkou *Telefonní seznam zákazníků* přes cizí klíč *id_tel_seznam_zak*.

Tabulka 5	**Zakaznik**
id zakaznika – pk	integer not null
Jmeno	varchar (15) not null
Prijmeni	varchar (30) not null
Ulice	varchar (40) null
cislo popisne	integer not null
Mesto	varchar (40) not null
PSC	integer not null
id smlouvy – ck	integer not null

Tabulka 5: *Zákazník*
Zdroj: Vlastní

V tabulce *Telefon* jsou uložena čísla jak prodejců, tak i zákazníků, čili je to telefonní seznam.

Tabulka 6	**Telefon**
id telefon – pk	integer not null
Telefon	varchar (13) not null

Tabulka 6: *Telefon*
Zdroj: Vlastní

V tabulce *Smlouva* jsou uloženy informace o konkrétní smlouvě. Na tabulky *Poznamky* a *CD Dokumenty* je relačně vázána přes cizí klíče *id_poznamky* a *id_cd_dokumenty*.

Tabulka 7	**Smlouva**
id smlouvy – pk	integer not null
EAN	bigint not nul
EIC	bigint not nul
Spotřeba	numeric (5,2) not null
Dtum odeslání	date not null
Dtum zneplatnění	date null
RWE karta	char (1) not null
SIM karta	char (1) not null
id poznamky – ck	integer not null
id cd_dokumenty - ck	integer not null

Tabulka 7: *Smlouva*
Zdroj: Vlastní

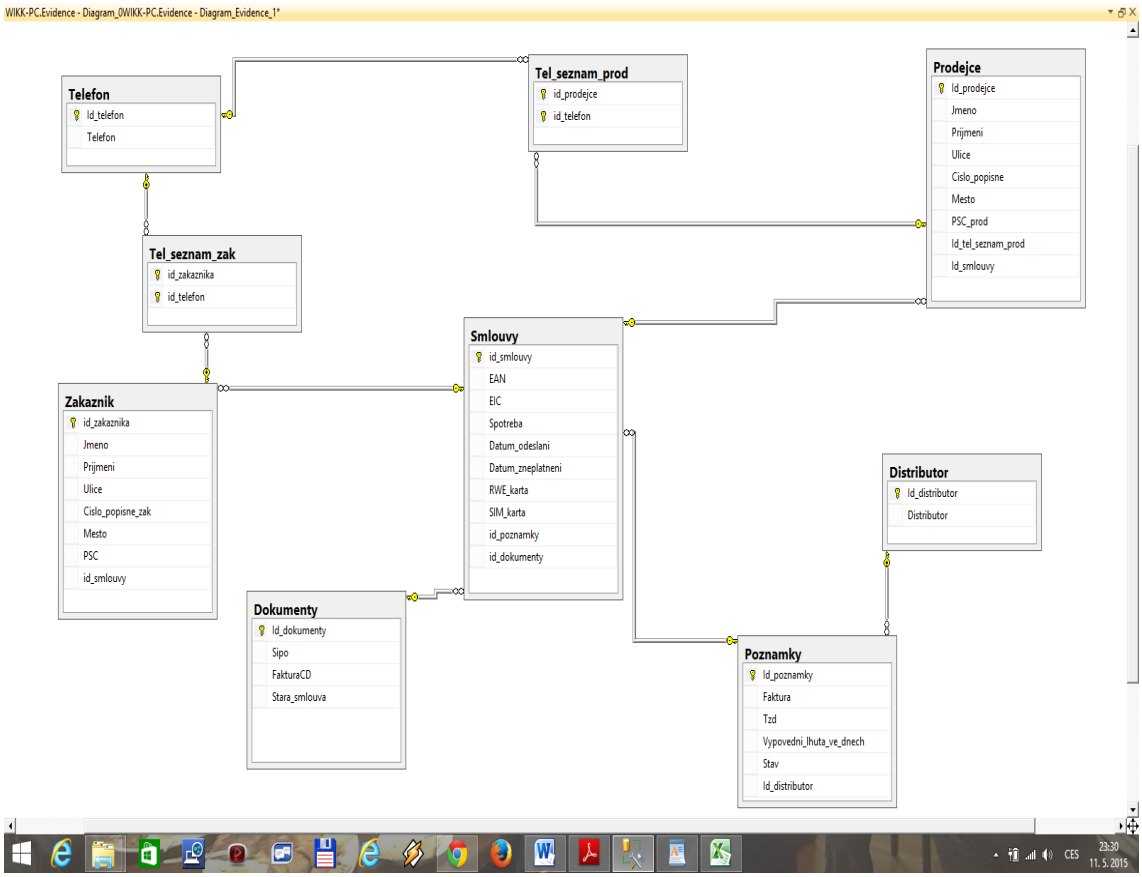
Tabulky 8 a 9 slouží pro dekompozici relací typu M:N mezi tabulkami *Telefon* a *Zákazník* a *Prodejce*. Oba cizí klíče v tabulkách dohromady tvoří i primární klíč dané tabulky

Tabulka 8	**Telefonní seznam prodejce**
id prodejce - ck – pk	integer not null
id telefon - ck – pk	integer not null

Tabulka 8: *Telefonní seznam prodejců*
Zdroj: Vlastní

Tabulk 9	**Telefonní seznam zákazníka**
id zakaznika	integer not null
id telefon - ck – pk	integer not null

Tabulka 9: *Telefonní seznam zákazníků*
Zdroj: Vlastní



Obrázek 20: *Databázový diagram z prostředí MS SQL Serveru Management Studia 2012*
Zdroj: Vlastní

Na obrázku 20 vidíme finální verzi vytvořenou během fází tvorby konceptuálního a logického modelu. Jsou pojmenovány všechny entity (tabulky), jejich atributy (sloupce tabulek), jsou normalizovány, odstraněny redundance a určeny všechny primární i cizí klíče. Nyní můžeme postoupit do další fáze.

3.5. fyzický model

V této fázi práce již přistoupíme k fyzickému vytvoření databáze pomocí jazyka SQL nebo Microsoft SQL Server 2012 přes nástroj *SQL Server Management Studio*. Na řádcích níže uvádím skript z tohoto prostředí, ve kterém je zaznamenána tvorba databáze evidence smluv.

```
CREATE DATABASE Evidence;

USE Evidence
go

CREATE TABLE Distributor
(
Id_distributor  integer not null,
Distributor          varchar (10) not null,
Primary Key (Id_distributor)
);

CREATE TABLE Poznamky
(
Id_poznamky          integer not null,
Faktura              char (1) Default 'N' not null
                     Check (Faktura IN ('A','N')),
Tzd                  Date null,
Vypovedni_lhuta_ve_dnech integer null,
Stav            char (11) null,
Id_distributor  integer not null
Primary key          (Id_poznamky),
Foreign key          (Id_distributor)
     references Distributor (Id_distributor)
);

CREATE TABLE Dokumenty
(
Id_dokumenty    integer not null,
Sipo            char (1) null
                     Check ( Sipo IN ('A','N')),
FakturaCD       char (1) null
                     Check ( FakturaCD IN ('A','N')),
Stara_smlouva   char (1) null
                     Check ( Stara_smlouva IN ('A','N')),
Primary key          (Id_dokumenty)
);

CREATE TABLE Telefon
(
Id_telefon      integer not null,
Telefon              varchar (13) not null,
Primary key          (Id_telefon)
);

CREATE TABLE Prodejce
(
```

```
Id_prodejce           integer not null,
Jmeno           varchar (15) not null,
Prijmeni        varchar (30) not null,
Ulice           varchar (40) null,
Cislo_popisne_prod    integer    not null,
Mesto           varchar (40) not null,
PSC_prod        integer not null,
Id_tel_seznam_prod    integer not null,
Id_smlouvy      integer not null,
Primary key           (Id_prodejce),
Foreign key           (Id_smlouvy)
     references Smlouvy (Id_smlouvy)
);

CREATE TABLE Smlouvy
(
id_smlouvy      integer not null,
EAN                   bigint not null,
EIC                   bigint not null,
Spotreba        numeric (5,2) not null,
Datum_odeslani  date not null,
Datum_zneplatneni date not null,
RWE_karta       char (1) default 'N' not null
                      Check ( RWE_karta IN ('A','N')),
SIM_karta       char (1) default 'N' not null
                      Check ( SIM_karta IN ('A','N')),
id_poznamky           integer not null,
id_dokumenty    integer not null,
Primary key           (id_smlouvy),
Foreign key           (id_poznamky)
     references Poznamky (id_poznamky),
Foreign key           (id_dokumenty)
     references Dokumenty (id_dokumenty)
);

CREATE TABLE Zakaznik
(
id_zakaznika    integer not null,
Jmeno           varchar (15) not null,
Prijmeni        varchar (30) not null,
Ulice           varchar (40) null,
Cislo_popisne_zak integer not null,
Mesto           varchar (40) not null,
PSC                   integer not null,
id_smlouvy      integer not null,
Primary key           (id_zakaznika),
Foreign key           (id_smlouvy)
```

```
     references Smlouvy (id_smlouvy)
);

CREATE TABLE Tel_seznam_prod
(
id_prodejce          integer not null,
id_telefon      integer not null,
Primary key          (id_prodejce, id_telefon),
Foreign key          (id_prodejce)
     references Prodejce (id_prodejce),
Foreign key          (id_telefon)
     references Telefon (id_telefon)
);

CREATE TABLE Tel_seznam_zak
(
id_zakaznika    integer not null,
id_telefon      integer not null,
Primary key          (id_zakaznika, id_telefon),
Foreign key          (id_zakaznika)
     references Zakaznik (id_zakaznika),
Foreign key          (id_telefon)
     references Telefon (id_telefon)
);
```

Tento skript přesně koresponduje s návrhem databáze z předcházejících fází. Nyní již mohou zaměstnanci firmy Ready Control s.r.o. vkládat data do této databáze.

3.6. pohledy a sestavy

V současné době ještě nejsou ještě zaměstnanci plně seznámení s prostředím, proto i vkládání dat přes příkaz INSERT provádím já. Pro vypisování dat využívají zaměstnanci velmi užitečný nástroj Management Studia, a to pomůcku pro vytváření SQL příkazů pro jednotlivé objekty. V podokně *Object Explorer* (nachází se zpravidla na levé straně okna) vyberete databázi a v ní databázovou tabulku, pro kterou chcete vytvářet šablonu příkazu. Vybírat můžete z těchto příkazů:

- CREATE TO
- DROP TO
- SELEC TO

- INSERT TO
- UPDATE TO
- DELETE TO

Například pro tabulku *Prodejce* z naší databáze *Evidence* vygeneruje pro příkazy INSERT a SELECT tyto šablony:

INSERT

```
USE [Evidence]
GO

INSERT INTO [dbo].[Prodejce]
           ([Id_prodejce]
           ,[Jmeno]
           ,[Prijmeni]
           ,[Ulice]
           ,[Cislo_popisne]
           ,[Mesto]
           ,[PSC_prod]
           ,[Id_tel_seznam_prod]
           ,[Id_smlouvy])
     VALUES
           (<Id_prodejce, int,>
           ,<Jmeno, varchar(15),>
           ,<Prijmeni, varchar(30),>
           ,<Ulice, varchar(40),>
           ,<Cislo_popisne, int,>
           ,<Mesto, varchar(40),>
           ,<PSC_prod, int,>
           ,<Id_tel_seznam_prod, int,>
           ,<Id_smlouvy, int,>)
GO
```

SELECT

```
USE [Evidence]
GO

SELECT [Id_prodejce]
      ,[Jmeno]
```

```
      ,[Prijmeni]
      ,[Ulice]
      ,[Cislo_popisne]
      ,[Mesto]
      ,[PSC_prod]
      ,[Id_tel_seznam_prod]
      ,[Id_smlouvy]
  FROM [dbo].[Prodejce]
GO
```

Stejně tak můžeme v MS SQL 2012 využít *Template Explorer* z menu *View*. V okně vpravo nahoře se nám otevře okno *Template Browser*, kde jsou uložené procedury, pomocí kterých můžeme vytvářet další šablony pro námi vytvořenou databázi *Evidence*.

Tímto způsobem můžeme vytvářet šablony pro *Spouště, Pohledy* apod..

3.7. Bezpečnost

Nesmíme samozřejmě zapomenout také na bezpečnost, a to ať už na ochranu dat před jejich ztrátou (například lidskou chybou), tak i před vniknutím do databáze nežádoucími subjekty (například hackeři, průmyslová špionáž a další) a jejich zneužitím. Já jsem tuto situaci řešil přes adresář *Security* na kartě *Object Explorer* vytvořením nové role *Uživatel*, které jsem přiřadil oprávnění vkládat data a upravovat data. Databázi jsem úmyslně koncipoval tak, aby nebylo potřeba mazat jakákoli data do ní vložená. A to proto, aby nemohlo omylem dojít ke smazání dat k tomu neurčených. Což je samozřejmě ještě pojištěno zálohou dat.

Přidat uživatele je možné i příkazem:

```
CREATE USER [Management] FOR LOGIN Manager1;
GO
```

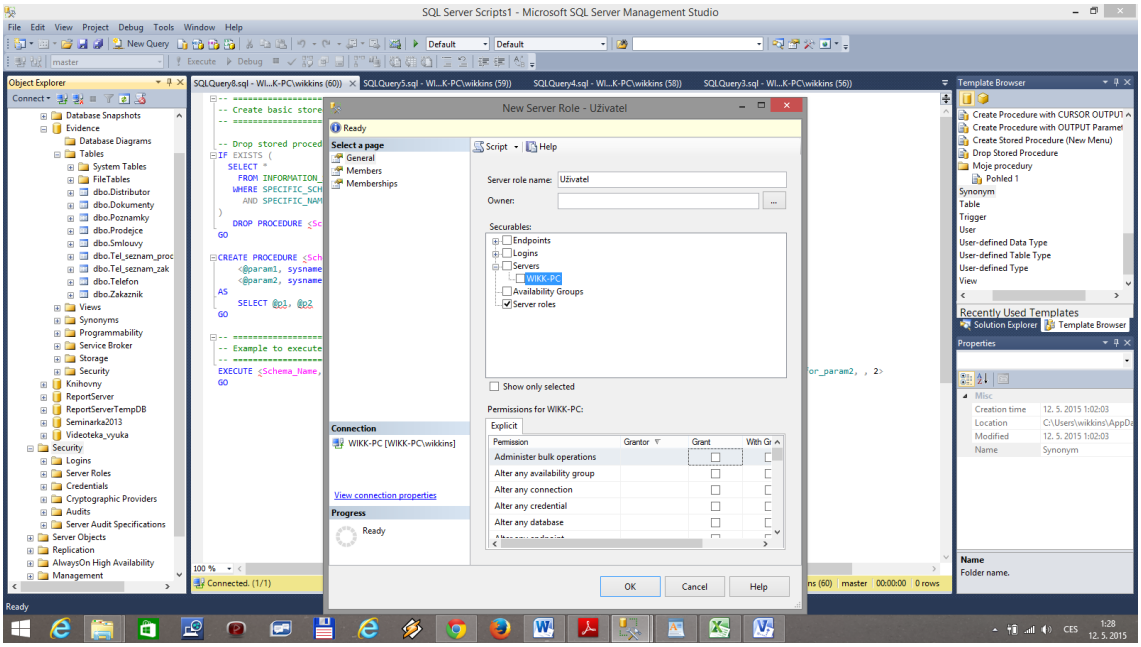
Ale lepší je vytvořit roli příkazem:

```
CREATE ROLE Uzivatel;
```

A aby to mělo smysl, přiřadíme *Uživateli* i určitá práva příkazem:

```
GRANT CREATE VIEW, INSERT, UPDATE, SELECT TO Uzivatel;
```

Já jsem to však vyřešil přes *Security – New Server role*.



Obrázek 21: *Dialogové okno pro vytvoření nové role – konkrétně Uživatele*
Zdroj: Vlastní

3.8. Přínosy

Cílem této práce byl pouze návrh evidence smluv, zatímco finální řešení ještě počítá s aplikací, která by měla usnadnit všem pověřeným uživatelům obsluhovat databázový systém, aniž by museli do hloubky studovat prostředí a tvorbu databází.

Prozatím firma investovala 23 774 Kč za zlevněnou verzi Microsoft SQL Server 2012 Standard edition, do současné doby (11.5.2015) 2000 Kč do školení jednoho zaměstnance (nicméně tato částka je vzhledem rozsahu školení téměř finální – max do 3000 Kč). Dále bude investovat do aplikace, přičemž její cenu momentálně neznám, ale měla by se pohybovat maximálně do 5000Kč.

Jediný finanční přínos spočívá v tom, že již nebude docházet k opakovaným vyplácením provizí za již jednou zaplacenou smlouvu. Byl to i hlavní důvod, proč management chtěl databázi evidence smluv vytvořit.

Dalšími přínosy bude také absolutní přehled o existujících, zaniklých smlouvách a kompletním přehledu o vývoji existujících smluv. Dále také budou veškera data aktuální, a to pro všechny uživatele, protože data budou soustředěny na jednom serveru.

Jakákoliv změna se tedy uloží přímo do databáze a ať přistoupíte z kteréhokoliv klienta instalovaného na počítačových stanicích, dostanete se aktuálním informacím. Odpadne tak složité přeposílání excelovských souborů, zjišťování zda je zrovna tenhle soubor aktuální, a jestli ta smlouva již existuje či ne. Což při existenci několika excelovských souborů, kde většina byla neaktuální, mohlo být velmi komplikované, pracné a náročné na čas.

4. Závěr

Cílem mé bakalářské práce bylo navrhnout databázi evidence smluv pro firmu Ready Control s.r.o.. Databáze měla obsahovat jednotlivé aktéry smluvních vztahů, tedy prodejce (obchodníka) a zákazníka. Konkrétní údaje o smlouvě a navíc měla tato databáze zachytit i „život“ těchto smluv. Znemožnit jejich opakované objevování se v jedné databázi. Zároveň evidovat smlouvy již zaniklé, které budou využívány pro následné analýzy.

Na základě popisu současné situace a analýzy problému jsem navrhl databázi, která již nyní splňuje všechny na ni kladené požadavky s tím, že do budoucna k ní bude ještě připojena webová aplikace, která uživatelům firmy ještě zjednoduší práci v databázovém prostředí.

Tato bakalářská práce splnila svůj cíl a navržená databáze by měla zjednodušit a zefektivnit práci úředníka, který se bude věnovat evidenci a kontrole smluv ve firmě Ready Control s.r.o..

Právě dalším a velmi důležitým krokem, který je nutný pro příjemné, pohodlné a efektivní využívání databáze, bude navržení a tvorba webové aplikace a uživatelského prostředí, a to v jazyce PHP. Tato aplikace bude pracovat nad databází a umožní mnohem širší využití databázového systému i Business Intelligence.

SEZNAM POUŽITÉ LITERATURY

(1) KŘÍŽ, Jiří. *Databázové systémy - DBS* [online]. Vysoké učení technické v Brně, [cit. 2014-22-05]. Dostupný z: http://luhan.comlu.com/DBS/doc/P/02/02.pdf

(2) GÁLA, Libor, Jan POUR a Zuzana ŠEDIVÁ. *Podniková informatika*. 2., přeprac. a aktualiz. vyd. Praha: Grada, 2009, 496 s. Expert (Grada). ISBN 978-80-247-2615-1.

(3) LACKO, Luboslav. *Business Intelligence v SQL Serveru 2008.* Brno: Computer Press, 2009. 456 s. ISBN 978-80-251-2887-9.

(4) LACKO, Ľuboslav. *Mistrovství v SQL Server 2012*. 1. vyd. Brno: Computer Press, 2013, 640 s. ISBN 978-80-251-3773-4

(5) BORONCZYK, Tim. *PHP 6, MySQL, Apache: vytváříme webové aplikace*. Vyd. 1. Brno: Computer Press, 2009, 816 s. ISBN 978-80-251-2767-4.

(6) KOCH, Miloš. *Datové a funkční modelování: vytváříme webové aplikace*. Vyd. 4., rozšířené. Brno: Akademické nakladatelství CERM, 2010, 142 s. Učební texty vysokých škol. ISBN 978-80-214-4125-5.

(7) OPPEL, Andrew J. *SQL bez předchozích znalostí: [průvodce pro samouky]*. Vyd. 1. Brno: Computer Press, 2008, 240 s. Učební texty vysokých škol. ISBN 978-80-251-1707-1.

(8) PROCHÁZKA, David. *PHP 6: začínáme programovat*. 1. vyd. Praha: Grada, 2012, 183 s. Průvodce (Grada). ISBN 978-80-247-3899-4.

(9) CÍSAŘ, Pavel. *InterBase/FireBird : podrobná příručka : tvorba, programování a správa databází.* 1. Vyd. Brno : Computer Press, 2003. 453 s. , 1 elektronický optický disk. ISBN 80-7226-946-1.

(10) KŘÍŽ, Jiří. *Databázové systémy - DBS* [online]. Vysoké učení technické v Brně, [cit. 2014-22-05]. Dostupný z: http://luhan.comlu.com/DBS/doc/P/04/04.pdf

Seznam obrázků

Seznam tabulek